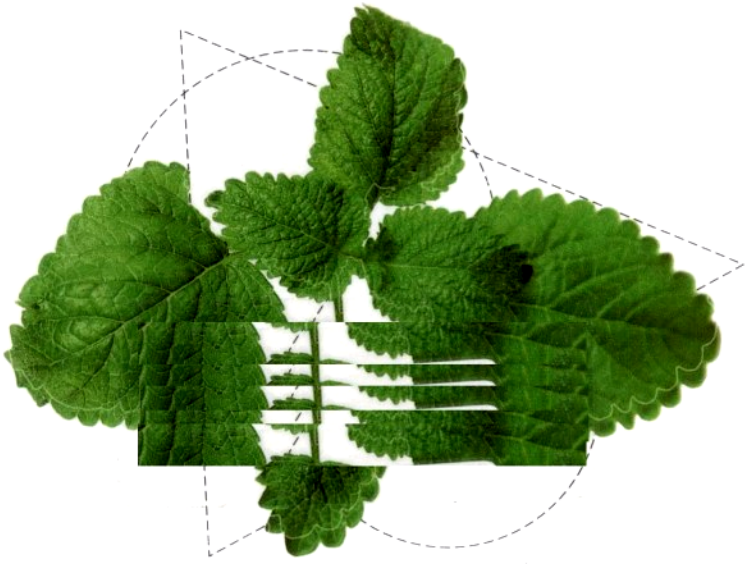
测试，还是测试！
我们介绍的不仅仅是工具，还有思想！

DVD-ROM
405分钟
超长
实战视频



LoadRunner

性能测试完全讲义

黄文高 赵 丹 编著

- 资深性能测试工程师深入剖析LoadRunner每个功能的“内幕”
- 全程实战，让你遨游于性能测试世界，体会性能测试全过程
- 超长语音视频讲解，教你如何玩转LoadRunner，同作者一起“悟道”
- 性能测试交流论坛（ www.testingba.com ）即将推出，敬请期待... ...

中国水利水电出版社
www.waterpub.com.cn

405分钟超长语音视频带你轻松实战性能测试！

LoadRunner
性能测试完全讲义

ISBN 978-7-5084-7457-1

销售分类：软件工程/软件测试

定价：38.00元（赠1DVD）

LoadRunner 性能测试完全讲义

黄文高　赵丹　编著

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

在软件测试行业，性能测试和自动化测试成为初级软件测试工程师迈向高级测试工程师必须跨越的一道门槛，而 LoadRunner 是性能测试工具的一面旗帜，谈到性能测试就不能不谈到 LoadRunner。

本书分三部分：入门篇、提高篇和实战篇。入门篇主要介绍性能测试基础知识、LoadRunner 基础知识和 LoadRunner 三大组件；提高篇采用大量的实例介绍 LoadRunner 的功能；实战篇使用两个案例就如何使用 LoadRunner 进行性能测试做了详细的介绍，从需求分析到结果分析都进行了详细的讲解，旨在通过案例分析功能或介绍使用技巧，希望读者能理解案例解决方案背后的思考过程、分析过程和推导过程。

本书适合暂时不了解性能测试，但又想了解性能测试、学习 LoadRunner 的读者；适合 LoadRunner 的初学者，希望看完本书他们能有很大的提高；适合中级性能测试工程师，希望本书的实践对他们的工作有益。

本书配套光盘中提供了书中实例所用脚本文件，PPT 电子讲义，以及书中内容的详细视频讲解，405 分钟超长视频与本书内容完美结合，深化 LoadRunner 的重点与难点，详细解剖 LoadRunner 的每个功能，带您轻松步入性能测试之路。

图书在版编目（CIP）数据

LoadRunner性能测试完全讲义 / 黄文高，赵丹编著
-- 北京 ：中国水利水电出版社，2010.5
ISBN 978-7-5084-7457-1

Ⅰ. ①L… Ⅱ. ①黄… ②赵… Ⅲ. ①性能试验－软件工具，LoadRunner Ⅳ. ①TP311.56

中国版本图书馆CIP数据核字(2010)第075040号

策划编辑：周春元　　责任编辑：李　炎　　封面设计：李　佳

书　　名	LoadRunner 性能测试完全讲义
作　　者	黄文高　赵丹　编著
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京市天竺颖华印刷厂
规　　格	184mm×240mm　16 开本　17 印张　403 千字
版　　次	2010 年 5 月第 1 版　2010 年 5 月第 1 次印刷
印　　数	0001—4000 册
定　　价	38.00 元（赠 1DVD）

凡购买我社图书，如有缺页、倒页、脱页的，本社营销中心负责调换

版权所有·侵权必究

前　言

目前，在国内，软件测试尚处于起步阶段，其市场表现实在有点令人尴尬。软件开发公司比比皆是，但软件测试公司却如凤毛麟角。市场化的第三方测试如同刚刚出生的宁馨儿，目前的市场几乎可以忽略不计。

国内IT企业也逐步开始重视对软件测试团队的建设，一些知名 IT 企业已经将软件测试作为企业未来发展的一个版块。目前在国内软件测试行业中，各种软件测试的方法、技术和标准都还在探索阶段。

有媒体报道过，国内开发人员与测试人员的比例是 8∶1，而国际公认的行业标准应该是 1∶1，可见国内测试行业与国外的差距比较大，这说明软件测试行业未来发展前景一片光明。

但是，从长远发展角度来看，测试还是需要高端人才。自动化测试、性能测试和对 Bug 的预防及早期检查是软件测试工程师面临的新的挑战。值得庆幸的是，近几年国内一些 IT 企业已经开始涉及自动化测试和性能测试。在软件测试行业，性能测试和自动化测试成为初级软件测试工程师迈向高级测试工程师必须跨越的一道门槛，随便打开一个软件测试的网站，都能看到性能测试的版块，而 LoadRunner 是性能测试工具的一面旗帜，谈到性能测试就不能不谈到 LoadRunner。

不论是性能测试，还是性能测试工具的杰出代表——LoadRunner，许多朋友在现实工作中并没有真正掌握。虽然会使用 LoadRunner 进行性能测试，但是测试结果并不能令人满意。笔者在工作中发现许多使用 LoadRunner 的人其实并不真正理解性能测试的意义，许多朋友不清楚如何分析性能测试需求？什么叫业务模型？什么叫场景模型？如何分析测试结果？笔者认为读者不应该仅仅满足于学会使用 LoadRunner，而应该能够从学习 LoadRunner 的过程中“悟道”。

从全国软件测试培训机构就 LoadRunner 培训内容调查，读者朋友最希望了解的内容是制定性能测试计划方案，进行性能测试需求分析、测试脚本开发、场景设计、测试执行及结果分析。故本书设计了两个不同架构的案例来分析使用 LoadRunner 进行性能测试的整个过程，不仅仅是讲述使用 LoadRunner 如何进行性能测试，更重要的目的是借案例来分析性能测试的思想，工具仅仅是载体，而思想才是最重要的。

本书结构清晰，内容安排由浅入深，对初学者来说可以很轻松的入门，当然对于有经验的读者朋友来说，可以有选择性的阅读。主要包括入门篇、提高篇和实战篇。

入门篇主要介绍的内容有：性能测试基础知识、LoadRunner 基础知识和 LoadRunner 三大组件。性能测试基础知识部分主要介绍了性能测试过程中一些常见的术语、性能测试划分和性能测试应用的领域；LoadRunner 基础知识部分主要介绍了 LoadRunner 的工作原理、工作过程和的内部结构，从全局的角度对 LoadRunner 工具进行了介绍；LoadRunner 三大组件部分主要介绍了 Vuser 发生器、Controller 控制器和 Analysis 分析器。在入门篇中只是简单介绍三大组件的常用操作及工作原理。

在提高篇中，将对 VuGen、Controller 控制器和 Analysis 分析器三大组件进行深入的了解，通过实例对测试过程中常用的技巧与方法进行详细的分析，就测试过程中容易出现错误的地方进行提示。

在实战篇中，选择两个案例就如何使用 LoadRunner 进行性能测试的整个过程进行全面的介绍。选择的两个实例的系统架构模式分别是 B/S 模式和 C/S 模式，目的是通过选取不同架构的系统进行实验。在实验过程中就这两种架构的系统在性能测试过程中的不同之处给予详细说明，并尽可能地在实战过程中引出更多的问题，希望给初学者以更多的帮助。

本书采用大量的实例和脚本对 LoadRunner 的使用进行介绍，从提高篇开始，本书就使用了大量的案例，目的是通过案例分析测试过程中常用的技巧，并且就测试过程中容易出错的地方进行提示，让读者朋友不仅仅是在看书，更重要的是在动脑思考。实战篇介绍的不仅仅是如何使用 LoadRunner，更重要的是分析性能测试的整个过程。最后两章使用两个案例讲解如何使用 LoadRunner 进行性能测试，从需求分析到结果分析都进行了详细的介绍。

本书中某些实例或许正好与读者面临的问题相同或存在相似之处，读者可以依葫芦画瓢地去解决现实中的问题，然而这并非作者的本意。作者的初衷是将思考方法和分析过程通过实例的方式传达给读者并与读者进行交流，希望读者能理解案例解决方案背后的思考过程、分析过程和推导过程。如果读者经过思考得出与作者不同的分析结果，或是证明作者所给出的解决方案并非最好的方案，这也是作者所期望的。

笔者希望读者在阅读本书的过程中，认真思考案例中如何进行需求分析、如何建立业务模型、如何建立场景模型和分析测试结果的过程，并将这些方法应用到现实的工作中去，而不要沉迷于书中给出的具体案例。另外，作者的分析方法和推导过程只是作者本人在工作中自己总结出的经验，不是标准答案，更不是圣经。期望读者能够认真思考作者的这些经验，并结合自己的实际，总结出一套自己的方法。如果真是这样，作者的这些文字工作就真正劳有所值了。

经过一年多的努力，书稿终于完成，在这里我感谢那些曾经帮助、支持和鼓励过我的朋友。

感谢中国水利水电出版社周春元编辑的帮助。

感谢我的同事黄胜杰、陈志坚、梁会美、农莉、沈东雪、向小飞、白晓霞和我的朋友黄金、黄海平、包静、万俊杰帮忙审核书稿。

感谢我的妻子韦玉凤奉献的爱心、支持与鼓励，并将此书献给我即将出生的孩子。

感谢父亲和母亲这些年来对我的养育之恩。

由于笔者水平有限，很多内容是自己的经验总结，出现错误在所难免，欢迎广大读者批评指正。读者在阅读本书的过程中如有任何不清楚的问题和批评建议，可以发邮件到 huangwengao@sina.com，作者将尽力给您答疑解惑。

最后，感谢您购买此书，希望您在本书中能够找到那些正在困扰着您的问题的答案。祝大家阅读愉快。

黄文高

2010 年 2 月

目　　录

前言

第一部分　入门篇

第 1 章　性能测试基础知识 2
1.1　什么是软件的性能 2
1.2　性能测试相关术语 3
1.2.1　响应时间 3
1.2.2　并发用户数 4
1.2.3　吞吐量 4
1.2.4　吞吐率 5
1.2.5　TPS 5
1.2.6　点击率 5
1.2.7　资源利用率 5
1.2.8　性能计数器 6
1.2.9　思考时间 6
1.3　性能测试划分 6
1.3.1　负载测试 7
1.3.2　压力测试 7
1.3.3　配置测试 7
1.3.4　并发测试 7
1.3.5　可靠性测试 8
1.4　性能测试应用领域 8
1.4.1　能力验证 8
1.4.2　规划能力 8
1.4.3　性能调优 9
1.4.4　缺陷发现 9
第 2 章　LoadRunner 基础知识 10
2.1　LoadRunner 简介 10
2.2　LoadRunner 工作原理 11
2.3　LoadRunner 工作过程 12
2.4　LoadRunner 内部结构 13
2.5　LoadRunner 测试步骤 16
第 3 章　Vuser 发生器 18
3.1　脚本录制 18
3.1.1　如何选择协议 19
3.1.2　开始录制脚本 22
3.2　Recording Options 设置 24
3.2.1　Recording 选项卡 24
3.2.2　Advanced 选项卡 26
3.2.3　Correlation 选项卡 27
3.3　Run-Time Settings 设置 27
3.3.1　Run Logic 选项卡 28
3.3.2　Pacing 选项卡 29
3.3.3　Think Time 选项卡 29
3.3.4　Miscellaneous 选项卡 30
3.4　脚本完善 31
3.4.1　插入事务 31
3.4.2　插入集合点 33
3.4.3　插入注释 34
第 4 章　Controller 控制器 36
4.1　场景类型介绍 36
4.1.1　手动测试场景 36
4.1.2　面向目标测试场景 38
4.2　场景设计 39
4.2.1　手动场景 Schedule 配置 39
4.2.2　面向目标场景 Schedule 配置 43
4.2.3　配置 View Script 47
4.2.4　配置 Load Generator 47
4.3　场景执行 48

4.3.1 场景控制 ····· 48
4.3.2 场景执行期间查看场景 ····· 53
4.4 场景监视 ····· 57
4.4.1 关于联机监控 ····· 57
4.4.2 监控器与度量 ····· 58
4.4.3 联机监视器 ····· 62
第 5 章 Analysis 分析器 ····· 66
5.1 Analysis 简介 ····· 66
5.1.1 Analysis 基础知识 ····· 66
5.1.2 设置选项 ····· 67
5.1.3 Analysis 图 ····· 72
5.2 摘要报告 ····· 74
5.2.1 概要部分 ····· 75
5.2.2 统计部分 ····· 75
5.2.3 事务统计部分 ····· 76
5.2.4 HTTP 响应统计 ····· 77
5.3 Analysis 常见图分析 ····· 79
5.3.1 Vuser 图 ····· 79
5.3.2 每秒点击数图 ····· 80
5.3.3 平均事务响应时间图 ····· 80
5.3.4 吞吐量图 ····· 81
5.4 Analysis 报告 ····· 82
5.4.1 HTML 报告 ····· 82
5.4.2 Word 报告 ····· 82
5.4.3 水晶报表 ····· 85

第二部分 提高篇

第 6 章 脚本编写 ····· 92
6.1 检查点 ····· 92
6.1.1 插入检查点 ····· 93
6.1.2 检查点函数 ····· 99
6.2 Block（块）技术 ····· 101
6.3 参数化技术 ····· 106
6.3.1 创建参数 ····· 106
6.3.2 参数类型属性 ····· 108
6.3.3 数据文件 ····· 108
6.3.4 导入数据 ····· 114
6.4 关联技术 ····· 118
6.4.1 录制中关联 ····· 119
6.4.2 录制后关联 ····· 121
6.4.3 手动关联 ····· 123
第 7 章 场景设计实践 ····· 127
7.1 集合点设置 ····· 127
7.2 IP 欺骗技术 ····· 130
7.2.1 IP Spoofer 设置 ····· 131
7.2.2 Controller 中启动 IP Spoofer ····· 134
7.3 负载均衡技术 ····· 136
7.4 执行路径转换 ····· 138
7.4.1 路径转换介绍 ····· 138
7.4.2 编辑路径转换表 ····· 139
7.5 在 LoadRunner 中使用功能测试脚本 ····· 141
7.5.1 QuickTest 创建 GUI Vuser 脚本 ····· 142
7.5.2 WinRunner 创建 GUI Vuser 脚本 ····· 143
7.5.3 场景中使用 GUI Vuser 脚本 ····· 144
第 8 章 结果分析实践 ····· 146
8.1 分析图合并 ····· 146
8.1.1 分析图合并原理 ····· 146
8.1.2 实例讲解 ····· 148
8.2 分析图关联 ····· 150
8.2.1 分析图关联原理 ····· 150
8.2.2 实例讲解 ····· 152
8.3 页面细分 ····· 154
8.3.1 页面细分原理 ····· 154
8.3.2 实例讲解 ····· 157
8.4 钻取技术 ····· 159
8.4.1 钻取技术原理 ····· 159
8.4.2 实例讲解 ····· 160

8.5 导入外部数据 161
8.5.1 导入数据工具 161
8.5.2 自定义文件格式 164
第 9 章 特殊协议 167
9.1 Windows Sockets（WinSock）协议 167
9.1.1 Windows Sockets 录制选项设置 168
9.1.2 Windows Sockets 录制 169
9.1.3 Windows Sockets 数据操作 173
9.1.4 关于 LRS 函数 177
9.2 邮件服务协议 182
9.2.1 邮件服务协议简介 182
9.2.2 邮件服务协议录制 184
9.2.3 脚本分析 190
9.2.4 关于 SMTP 和 POP3 函数 192

第三部分 实战篇

第 10 章 客户关系管理系统性能测试 196
10.1 系统介绍 196
10.2 需求分析 198
10.2.1 性能指标 198
10.2.2 需求详细分析 198
10.3 测试方案及计划 200
10.3.1 人力资源 200
10.3.2 时间进度 200
10.3.3 测试环境准备 200
10.3.4 业务模型创建 201
10.3.5 场景模型创建 202
10.3.6 测试数据准备 203
10.4 测试用例 204
10.5 执行测试 207
10.5.1 脚本开发 207
10.5.2 场景设计 212
10.5.3 计数器设置 217
10.5.4 场景监视 219
10.6 结果分析 221
10.7 测试结论 229
第 11 章 信息系统性能测试 230
11.1 系统介绍 230
11.2 需求分析 231
11.2.1 性能指标 231
11.2.2 需求详细分析 231
11.3 测试方案及计划 232
11.3.1 人力资源 232
11.3.2 时间进度 232
11.3.3 测试环境准备 233
11.3.4 业务模型创建 234
11.3.5 场景模型创建 234
11.3.6 测试数据准备 235
11.4 测试用例 236
11.5 执行测试 237
11.5.1 脚本开发 237
11.5.2 场景设计 241
11.5.3 计数器设置 247
11.5.4 场景监控 247
11.6 结果分析 249
11.7 测试结论 257
附录 A 主要计数器 258
附录 B 性能测试 j 模型 262

第一部分

入门篇

入门篇主要介绍的内容有：性能测试基础知识、LoadRunner 基础知识和 LoadRunner 的三大组件。性能测试基础知识部分主要介绍了性能测试过程中一些常见的术语、性能测试划分和性能测试应用的领域；LoadRunner 基础知识部分主要介绍了 LoadRunner 的工作原理、工作过程和内部结构，从全局的角度对 LoadRunner 工具进行介绍；LoadRunner 的三大组件部分主要介绍了 Vuser 发生器、Controller 控制器和 Analysis 分析器。入门篇中只是简单介绍三大组件的常用操作及工作原理，其使用技巧和高级使用将在提高篇中进行详细的介绍。

1

性能测试基础知识

1.1 什么是软件的性能

一般来说，性能是一种指标，表明软件系统或构件对其及时性要求的符合程度；其次，性能是软件产品的一种特性，可以用时间来进行度量。

性能的及时性用响应时间或吞吐量来衡量。响应时间是指服务器对请求作出响应所需要的时间。

对于单个事务，响应时间是指完成事务所需的时间；对于用户任务，响应时间体现为端到端的时间。比如，“用户点击‘确定’按钮后 3 秒内呈现出结果”就是一个对用户任务响应时间的描述，在这个用户任务中，可能有多个具体的事务需要完成，每个事务都有其单独的响应时间。

对交互式应用（如 Web 应用）来说，一般以用户感受到的响应时间来描述系统的性能，而对非交互式应用（如嵌入式系统或银行等业务处理）而言，响应时间是指系统事件产生响应所需的时间。

通常，关注软件性能的对象是多个层面的，用户关注软件性能，系统管理员关注软件性能，软件开发工程师也关注软件性能，下面从三个不同的层面对软件性能做一个简要的介绍。

1. 用户

从用户的角度来说，软件性能是软件对用户操作的响应时间。通俗的讲，如果用户点击一个提交或输入一个 URL 地址，随后系统把结果呈现到用户眼前，这个过程所花费的时间即为用户对软件性能的直观印象，如图 1-1 所示。

需要注意的一点是，用户体会到的“响应时间”，既有客观的成分，也有主观的成分，从提交业务到系统开始返回信息的时间被用户认为是系统的响应时间，例如，用户执行某个操作，该操作会返回大量的数据，假如所有的数据并不会同时返回，而是先将一部分数据呈现出来，再将全部数据呈现出来，此时，用户体会的响应时间为从执行操作到有部分数据呈现出来的时间，而真正的响应时间应该是系统将全部数据呈现出来的时间。

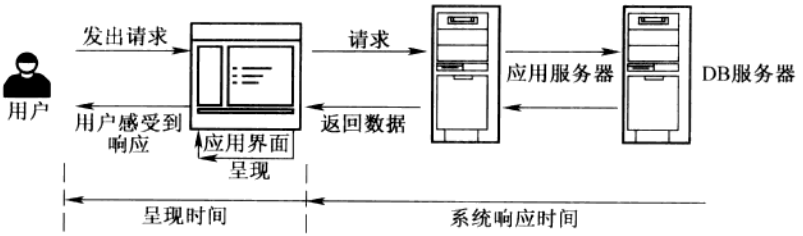


图 1-1 Web 系统响应

2. 系统管理员

从系统管理员的角度来说，软件性能在响应时间方面与用户的视角是一样的。但管理员是一群特殊的用户群体，管理员除关注一般用户体验外，还关注与系统状态相关的信息。如系统资源的使用情况，包括 CPU 的使用情况、内存的使用情况、磁盘 I/O 等，当然还有数据交互的情况。

另外，管理员还关注系统硬件资源的可扩展性即规划性能部分。比如，系统现在支持 100 个用户并发没问题，那么将来支持 200 个用户并发时是否会出现性能问题呢？

3. 软件开发工程师

从软件开发工程师的角度来说，他们关注用户和管理员关注的所有问题。另外还关注内存泄漏、数据库是否出现死锁、中间件以及应用服务器等问题。

1.2 性能测试相关术语

本小节将介绍性能测试过程中的一些相关术语：响应时间、并发用户数、事务响应时间、吞吐量、吞吐率、TPS（每秒事务响应数）、性能计数器等。

1.2.1 响应时间

响应时间是指应用系统从发出请求开始到客户端接收到所有数据所消耗的时间。该定义强调所有数据都已经被呈现到客户端所花费的时间，为什么说是所有数据呢？因为用户体验的响应时间带有主观性，用户认为从提交请求到服务器开始返回数据到客户端的这一段时间为响应时间。

现在对响应时间进行细分，以一个 Web 应用的页面响应时间为例，如图 1-2 所示。从图中可以看到，页面的响应时间可分解为“网络传输时间”（N1+N2+N3+N4）和“应用延迟时间”（A1+A2+A3），而“应用延迟时间”又可分解为“数据库延迟时间”（A2）和“应用服务器延迟时间”（A1+A3）。

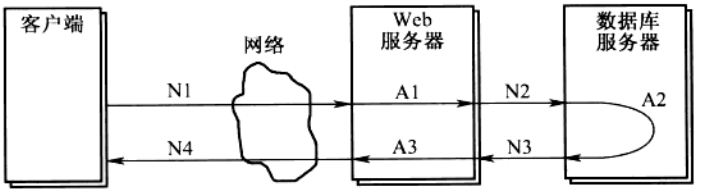


图 1-2 Web 页面响应时间分解

1.2.2 并发用户数

并发用户数指同一时刻与服务器进行数据交互的所有用户数量。概念中有两点需要注意。第一：同一时刻，因为并发强调的是用户同时对服务器进行施压。例如：一个人同时挑两件东西，这时表示两件东西同时被这个人挑起来，而如果是先挑一件，再挑另外一件，那么就无法表现出同时的概念，这两件东西也就没有同时施压在这个人身上。第二：强调要与服务器进行数据交互，如果未和服务器进行数据的交互，这样的用户是没给服务器带来压力的。同样是上面的例子，这个人虽然同时挑了两件东西，但其中有一件东西是没重量的，那就是说只有一件东西对这个人造成压力。

因此对于并发用户这个概念的理解经常会出现以下两种误区：一是认为系统所有的用户都叫并发用户；二是认为所有在线的用户都是并发用户。在线用户不一定是并发用户，原因是在线用户不一定就与系统进行了数据的交互，例如：如果一些在线用户只是查看系统上的一些消息，那么这些在线用户不能作为并发用户计算，因为这些用户并没有与系统进行数据交互，不会给服务器带来任何压力。

那么并发用户数如何计算呢？目前并没有一个精确的计算公式，很多情况下都是根据以往的经验进行估算。根据行业的不同，并发用户数也会有所不同，像电信行业并发用户数为在线用户的万分之一，如果有 1000 万在线用户，那么需要测试 1000 个并发用户即可。OA（办公自动化）系统的并发用户数一般是在线用户的 5%～20%左右，所以并发用户数很大程度上是根据经验和行业的一些标准来计算的。

1.2.3 吞吐量

在性能测试过程中，吞吐量是指单位时间内服务器处理客户请求的数量，吞吐量通常使用请求数/秒来衡量，其直接体现服务器的承载能力。

吞吐量作为性能测试过程中主要关注的指标之一，它与虚拟用户数之间存在一定的联系，当系统没有遇到性能瓶颈时，可以采用下面这个公式来计算。

$F=\frac{N_{VU}\times R}{T}$ 其中，F 表示吞吐量；N_{vu} 表示 VU（Virtual User，虚拟用户数）的个数；R 表示每个 VU 发出的请求数量；T 表示性能测试所用的时间。但是如果系统遇到性能瓶颈，这个公式就

不再适用。吞吐量与 VU 之间的关联图如图 1-3 所示。从图中可以看出，吞吐量在 VU 增长到一定数量时，软件系统出现性能瓶颈，此时，吞吐量的值并不会随着 VU 数量的增加而增大，而是趋于平衡。

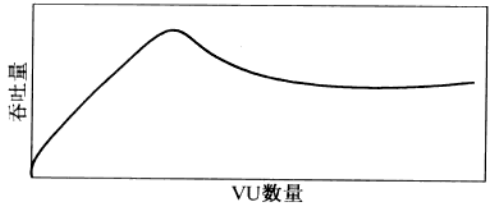


图 1-3 吞吐量-VU 数量关系图

1.2.4 吞吐率

吞吐率（Throughout）是指单位时间内从服务器返回的字节数，也可以指单位时间内服务器处理客户提交的请求数。它是衡量网络性能的一个重要指标。吞吐率=吞吐量/测试时间，通常情况下吞吐量的值越大，吞吐率的值也越大，吞吐率的值越大系统的负载能力越强。

1.2.5 TPS

TPS（Transaction Per Second）表示服务器每秒处理的事务数，它是衡量系统处理能力的重要指标。

1.2.6 点击率

点击率（Hit Per Second）是指每秒钟用户向服务器提交的 HTTP 数量。用户每点击一次，服务器端就要对用户提交的请求进行一次处理，从事务的角度来说，如果把每次点击作为一次提交事务来对待，那么点击率与 TPS 的概念是等同的。对于 Web 系统来说，“点击率”是服务器处理的最小单位，点击率的值越大，说明服务器端所能承受的压力越大。因此通常情况下，Web 服务器都具有防刷新的机制，因为客户每刷新一次系统就要响应一次点击，如果不对服务器进行防刷新处理，当用户不停地点击刷新按钮，此时服务器将承受巨大的压力。

需要注意的是，点击一次并不代表客户端只向服务器端发送一个 HTTP 请求，客户每点击一次，都会向服务器端发出多个 HTTP 请求。

1.2.7 资源利用率

资源利用率是指服务器系统中不同硬件资源被使用的程度，资源使用率=资源实际使用量/总的

可用资源量。主要包括 CPU 利用率、内存利用率、磁盘利用率、网络等。资源利用率是分析系统性能指标进而改善性能的主要依据，在配置调优测试的过程中，通过比较配置调优前后系统资源的利用率来判断调优的效果。

1.2.8 性能计数器

性能计数器（Counter）是描述服务器或操作系统性能的一些数据指标。主要是通过添加计数器来观察系统资源的使用情况。性能计数器包括操作系统性能计数器、数据库计数器、应用服务器计数器等。

计数器在性能测试过程中发挥着“监控和分析”的关键作用，尤其是在分析系统的可扩展性和对性能瓶颈进行定位时，计数器的阈值起着非常重要的作用。必须注意的是，一般情况下，单一的性能计数器只能体现系统性能的某一个方面，在性能测试过程中分析测试结果时，必须基于多个不同的计数器进行分析。

在性能测试中常用资源利用率进行横向对比。如在进行性能测试时发现，某个资源的使用率很高，几乎达到 100%，假设该资源是 CPU，而其他资源的使用率又比较低，这时可以很清楚地知道，CPU 是系统性能的瓶颈。

1.2.9 思考时间

思考时间（Think Time），也称为“休眠时间”，是指用户在进行操作时，每个请求之间的时间间隔。对于交互系统来说，用户不可能持续不断地发出请求，一般情况下，用户在向服务器端发送一个请求后，会等待一段时间再发送下一个请求。

在测试脚本中，思考时间为脚本中两条请求语句之间的间隔时间。当前对于不同的性能测试工具提供了不同的函数来实现思考时间，在实际的测试过程中，如何设置思考时间是性能测试工程师要关心的问题。

1.3 性能测试划分

性能测试划分有很多种，测试方法也有很多种，更确切的说是由于测试方法的不同决定了测试划分的情况，但在测试过程中性能测试的划分没有绝对的界限，常用的有压力测试、负载测试和并发用户测试等。

性能测试的方法主要包括以下几种：

- 负载测试（Load Testing）
- 压力测试（Stress Testing）
- 配置测试 (Configuration Testing)
- 并发测试（Concurrency Testing）
- 可靠性测试（Reliability Testing）

1.3.1 负载测试

负载测试（Load Testing）是通过对被测试系统不断地加压，直到超过预定的指标或者部分资源已经达到了一种饱和状态不能再加压为止。就像举重运动员，在举重的过程中不断地增加杠铃重量，直到运动员无法举起。

该方法主要是为了找到系统最大的负载能力，为性能调优提供数据。该测试方法有以下几个特点：

1）目的：找到系统最大的负载能力。

2）环境：该方法需要在特定的环境下进行测试。

3）手段：不断地对系统进行加压，直到系统中部分资源达到极限。

1.3.2 压力测试

压力测试（Stress Testing）是指系统已经达到一定的饱和程度（如 CPU、磁盘等已经处于饱和状态），此时系统处理业务的能力，系统是否会出现错误。

疲劳测试是压力测试的一种表现形式。例如，一个人很累了，但还在持续不停的工作。

该测试方法有以下几个特点：

1）目的：测试在系统已经达到一定的饱和程度时，系统处理业务的能力。

2）手段：使用模拟负载等方法，使系统资源达到一个较高的水平。

3）该方法一般用于系统稳定性测试。

1.3.3 配置测试

配置测试（Configuration Testing）是通过调整系统软/硬件环境，了解各种不同环境对系统性能的影响，从而找到系统的最优配置。

该测试方法有以下几个特点：

1）目的：通过调整环境了解不同因素对系统性能的影响情况，从而找到调优的方法。

2）手段：通过调整系统软/硬件环境，使系统在不同环境下进行性能测试。

3）该方法一般用于系统调优和规划能力。

1.3.4 并发测试

并发测试（Concurrency Testing）是通过模拟用户并发访问，测试多用户同时访问同一应用、模块或数据，观察系统是否存在死锁、系统处理速度是否明显下降等其他的一些性能问题。

该测试方法有以下几个特点：

1）目的：当多用户并发访问时，系统是否存在一些可能的并发问题。

2）手段：模拟多用户同时并发操作。

1.3.5 可靠性测试

可靠性测试（Reliability Testing）是当系统在一定的业务压力下，让系统持续运行一段时间，观察系统是否达到要求的稳定性，此处强调在一定业务压力下持续运行的能力，可靠性测试必须给出一个明确的要求，如系统能够持续无故障运行多少天。

该测试方法有以下几个特点：

1）目的：测试系统在一定的业务压力下，系统可持续运行的时间。

2）环境：指明系统在一定的业务压力环境下持续运行。

3）测试过程中要关注系统运行的情况。

1.4 性能测试应用领域

上一小节讲了常用的性能测试方法，本小节将从性能测试的应用领域来讲述性能测试的分类，从应用领域来划分，性能测试分为以下四大领域：

- 能力验证
- 规划能力
- 性能调优
- 缺陷发现

1.4.1 能力验证

能力验证是性能测试最常用的一个领域。一般能力验证采用这样的描述方式：“某系统能否在条件 A 下具备 B 性能”。重点在于验证系统是否具备某种能力。

能力验证领域有以下几个特点：

1）要求在一个已确定的环境下运行。

2）需要根据典型场景来设置测试方案与测试用例。

1.4.2 规划能力

规划能力与能力验证有相似之处，但还是存在一些不同的地方，能力验证强调的是在某个条件下具备什么样的能力，而规划能力体现系统如何才能达到要求的性能指标。规划能力问题常常会这样描述：“系统如何才能支持未来用户增长的需要”，这里强调的是未来能力增长的一个需求，着眼于未来系统的规划。

规划能力领域的特点是：

1）对系统能力的一种探索性的测试。

2）可以了解系统的性能及系统性能的可扩展性。

1.4.3 性能调优

性能调优是通过测试来调整系统的环境，最终使系统性能达到最优的状态。这是一个持续调优的过程，主要调优的对象有数据参数、应用服务器、系统的硬件资源等。

一个标准性能调优的步骤如图 1-4 所示。

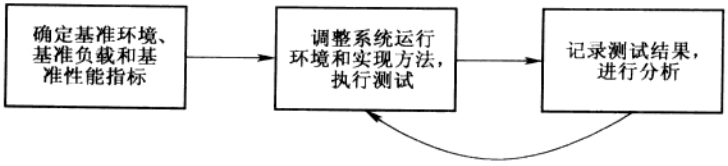


图 1-4 性能调优过程

1.4.4 缺陷发现

性能测试应用领域的主要目标是通过性能测试的手段来发现系统存在的缺陷。很多系统在实验室测试环境中没有任何问题，可是当交付给客户时就出现了莫明其妙的错误。如果交付给客户后出现多人同时访问速度缓慢或宕机的现象，那么很有可能是由于系统性能问题所引起。

2

LoadRunner 基础知识

2.1 LoadRunner 简介

LoadRunner 是一种预测系统行为和性能的负载测试工具。以模拟上千万用户并发负载并实时监测系统性能的方式来确认和查找问题。LoadRunner 能够对整个企业架构进行测试。通过使用 LoadRunner，企业能最大限度地缩短测试时间，优化性能和加速应用系统的发布周期。

目前企业的网络应用环境都必须支持大量用户同时访问，网络体系架构中包含各类应用环境及由不同供应商提供的软件和硬件产品，难以预知的用户负载和愈来愈复杂的应用环境使公司不得不担心会发生用户响应速度过慢、系统崩溃等问题，从而导致公司收益的损失。Mercury Interactive 的 LoadRunner 能让企业保护自己的收入来源，在无需购置额外硬件的情况下最大限度地利用现有的资源，并确保终端用户在应用系统的各个环节中对其测试应用的质量、可靠性和可扩展性都有良好的评价。

LoadRunner 具有以下几个特点：

1. 轻松创建虚拟用户

使用 LoadRunner 的 Virtual User Generator，能便捷地创立起系统负载。该引擎能够生成虚拟用户，以虚拟用户的方式模拟真实用户的业务操作行为。它先记录下业务流程（如下订单或机票预定），然后将其转化为测试脚本。利用虚拟用户，可以在 Windows、UNIX 或 Linux 机器上同时产生成千上万个用户访问。所以 LoadRunner 能极大地减少负载测试所需的硬件和人力资源。另外，LoadRunner 的 TurboLoad 专利技术能提供很高的适应性，TurboLoad 使您可以产生每天几十万名在线用户和数以百万的点击数负载。

2. 创建真实的负载

Virtual Users 建立后，需要确定负载方案、业务流程组合和虚拟用户数量；使用 LoadRunner 的 Controller，能很快组织起多用户的测试方案。Controller 的 Rendezvous 功能提供一个互动的环境，在这种环境下既能建立起持续且循环的负载，又能管理和驱动负载测试方案，而且还可以利用它的

日程计划服务来定义用户在什么时候访问系统以产生负载。完成上述步骤后，测试过程便自动化了。

3. 定位性能问题

LoadRunner 内含集成的实时监测器，在负载测试过程的任何时候，都可以观察到应用系统的运行性能。这些性能监测器将实时显示交易性能数据（如响应时间）和其他系统组件，包括 Application Server、Web Server、网络设备和数据库等的实时性能。这样，就可以在测试过程中从客户和服务器双方评估这些系统组件的运行性能，从而更快地发现问题。

4. 重复测试保证系统的高性能

负载测试是一个重复过程。每次处理完一个出错情况，都需要对应用程序在相同的方案下再进行一次负载测试，以检验所做的修改是否改善了运行性能。

2.2 LoadRunner 工作原理

第一章介绍了性能测试的一些基础知识，本小节主要讲述 LoadRunner 的工作原理，从宏观的角度了解 LoadRunner 的总体框架、LoadRunner 四大组件是如何工作的以及它们之间是如何通信的。

LoadRunner 由四大组件组成：VuGen、控制器、负载发生器和分析器。各组件之间的关系如图 2-1 所示。

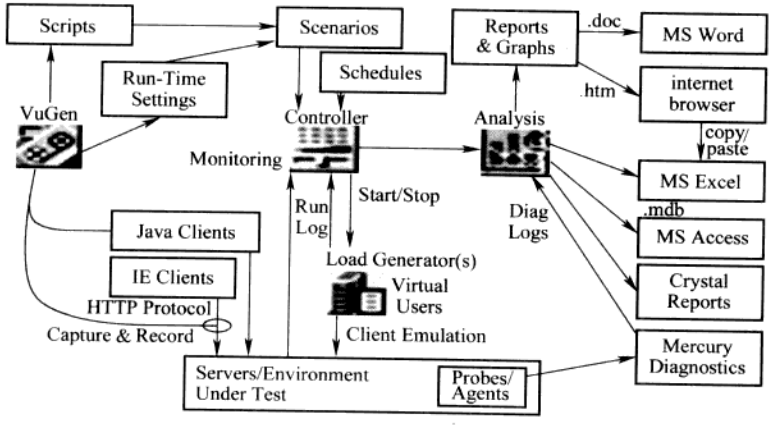


图 2-1 总体架构原理图

1）VuGen 发生器（Visual User Generator，VuGen）：它的作用是捕捉用户的业务流，并最终将其录制成一个脚本。在录制脚本前首先选择一种协议（脚本录制前如何选择协议的内容在后面章节中会详细讲述）；接着在客户端模拟客户实际使用过程中的业务流程，并录制成一个脚本；然后编辑脚本和设置 Run-Time Settings 项；最后 VuGen 通过对脚本的编译生成一个没有错误的可运行的脚本。

2）控制器（Controller）：控制器部分包含两大作用。第一：设计场景。脚本编辑完成后，必

须对脚本如何运行设计一种策略。场景设计主要包括手动场景设计和目标场景设计两种方式；第二：场景监控。控制器可以实时监控脚本运行的情况。可以通过添加计数器来监控 Windows 资源、应用服务器和数据库使用情况。

场景设计的目的是设计出一个最接近用户实际使用的场景，场景设计越接近用户使用的实际情况，测试出来的数据就越接近真实值。否则测试出来的数据是不可靠的，到系统上线时可能还是会出现性能问题，场景设计也涉及很多技巧，如 IP 欺骗、负载均衡等一些手段。

执行实时监控更多的是对资源使用情况的监控，以检测这些系统资源是否可能存在瓶颈。

3）负载发生器（Load Generators）：模拟用户对服务器提交请求。

正常情况下，在性能测试过程中会将控制器和负载发生器分开，即控制器使用一台独立的机器，为什么会这样做呢？因为在进行脚本编辑时会产生大量的参数化文件，而这些参数化文件会占用系统资源，再者就是运行时会产生大量的日志文件，所以在测试过程中一般都会将控制器与负载发生器分开。

负载发生器的计算：在测试时，需要计算测试过程中需要使用多少台负载发生器才算是合理的，任何一台计算机所能支持的虚拟用户数都有一个上限值，如果在测试过程中需要测试的并发虚拟用户数超过计算机所能支持的最大虚拟用户数，这时测试的负载机本身也就成为性能的瓶颈。每个虚拟用户运行时需要占用一定的系统内存资源，具体的值可以从官方文档中获得，通过这个值可以计算出一台计算机最多可以支持多少个虚拟用户。例如，假设负载发生器的计算机使用的内存容量为 512MB，在测试过程中每个虚拟用户需要的内存资源大概为 2.5MB，那么这台计算机最多只能支持 200 个虚拟用户并发测试，如果需要测试 500 个虚拟用户并发，那么就需要两台计算机。

需要注意的是当使用多台负载发生器时，一定要保证负载均衡。负载均衡是指在进行性能测试的过程中，保证每台负载发生器均匀地对服务器进行施压。如果负载处于不均衡的情况，那么在测试过程中，有的负载机很忙，而有的负载机又处于很闲状态，这样测试出来的值是不可靠的。

4）分析器（Analysis）：一个数据分析工具，主要用于对测试结果进行分析。Analysis 分析器中提供了很多基础的数据，但是仅仅依靠这些基础的数据是不够的，客户看到这样的报告也不会满意。在这里涉及到很多分析技术，常用的分析技术有：合并、叠加、页面细分和钻取技术等，这些技术在后面会进行详细的描述。Analysis 的另一个优点就是它本身提供了很多报告的形式，包括 XML、Word 等，这是 LoadRunner 做得比较出色的一部分。

以上是 LoadRunner 四大组件的作用和特点。

2.3 LoadRunner 工作过程

LoadRunner 的四大组件有着各自的作用和特点，但只有将这四大组件联系起来，才能更好地去完成性能测试工作，如图 2-2 所示。

第一步：通过 VuGen 来设计脚本。包括录制脚本和编辑脚本。但在录制脚本之前需要分析业务模式并建立业务模式（通过对业务模式的分析可使模拟过程更接近用户的实际使用）。脚本录制

完成之后，需要对脚本进行编辑，主要是对脚本进行修改和增强。

第二步：通过 Controller 来设计场景和执行场景。在场景设计之前必须对业务环境的场景进行充分的分析，然后新建场景，这样才能设计出更接近用户实际使用的场景模型。接下来要执行场景，Controller 通过加载脚本的形式来控制虚拟用户脚本运行的情况以保证虚拟用户按录制时的业务流程来执行。

第三步：虚拟用户并发执行。Controller 控制器控制着脚本运行时的负载机（也就是虚拟用户），虚拟用户是通过负载机产生的，在场景设计时其中有一项便是用来设计负载发生器的，只有设计好这项内容 Controller 才可以调度负载机。

一般情况会错误地认为没有负载机，这是因为在测试过程中将 Controller 和负载机设置为同一台机器，而在实际测试过程中并不允许这样做，因为 Controller 控制机在录制和编辑脚本时会占用系统资源，故一般都会将两者分别放在不同的计算机上来执行。

第四步：结果分析。当场景执行测试结束后，会生成一些分析结果的数据，这时测试工程师需要对这些数据进行分析，如果结果能满足需求，那么说明系统性能满足需求，反之，就有可能需要多次进行实验，来找到性能瓶颈并向开发工程师提出解决的建议和性能调优的建议。

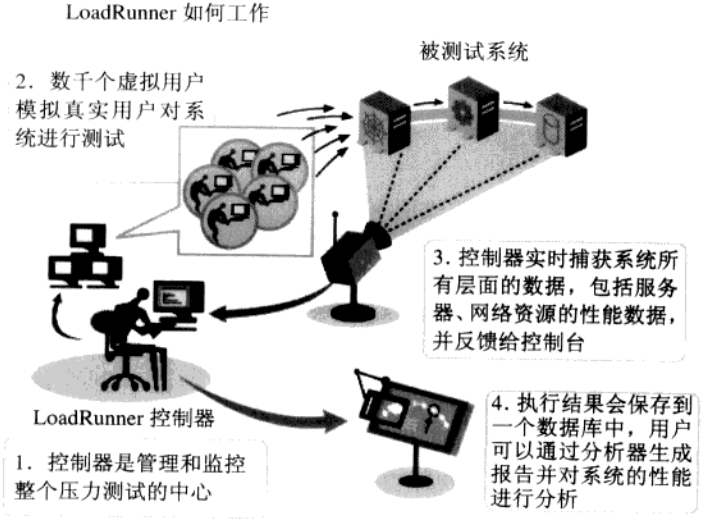


图 2-2 LoadRunner 工作过程

从图 2-2 中可以很清晰地看到四大组件如何相互协调来完成性能测试工作的。

2.4 LoadRunner 内部结构

LoadRunner 主要通过控制内部程序的调度来控制整个性能测试过程，LoadRunner 内部结构图

如图 2-3 所示。该图详细地描述了 LoadRunner 执行过程中内部程序是如何调度的及内部各程序之间的关系。

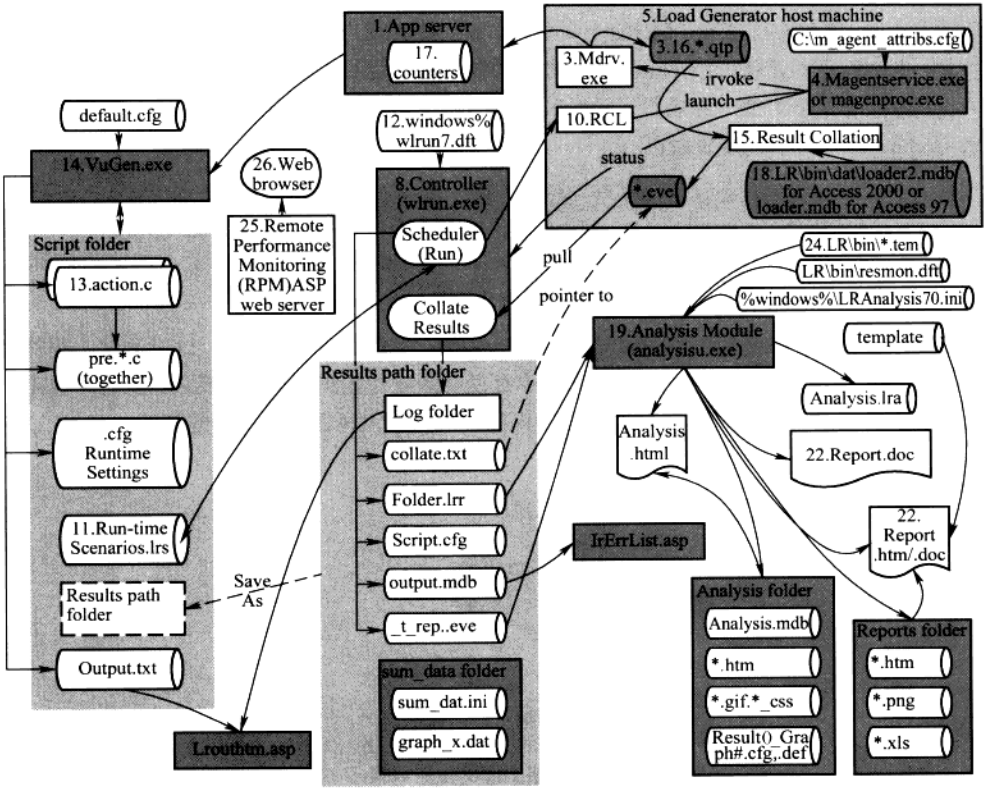


图 2-3 LoadRunner 内部结构图

下面从 LoadRunner 内部结构的层次来分析 LoadRunner 性能测试的过程。

1）首先准备好待测试的应用服务器和待测试的系统。

2）LoadRunner 中多线程驱动进程 mdrv.exe 和 r3vuser.exe 模拟产生压力，其中 r3vuser.exe 仿真应用程序的客户端，如 IE 浏览器。它执行了以下三个主要的操作：

①cci（C 语言编译器）建立 ci 文件，然后使用被测系统的协议来执行。

②通过 Windows 批处理脚本启动 mdrv.exe 程序从而启动 LoadRunner 的运行。mdrv 能自动停止加载 Vuser，因为它们与 Vuser 和 Windows 负载发生器上的 CPU 监视器之间互相通信。

③在 Windows 机器上，对于每一个基于 Java 的 Vuser 都有一个独立的 JVM，注意 UNIX 平台不支持 Java Vuser。

3）虚拟用户在负载发生器端的计算机上使用代理作为服务或进程时，按照组启动方式启动虚

拟用户，用户组是多个 Vuser 组成的逻辑集合，在 Vuser 发生器上运行相同的脚本。

4）每个负载发生器（Load Generator）都维护着一个以 qtp 为后缀名的执行日志。

5）日志服务启动后，代理会根据用户组进行隔离，在结果文件中为每个虚拟用户建立一个顺序文件。

6）在执行过程中，这些文件会在“视图”→“显示”输出窗口中显示出来。

7）在预先设置延时上，Controller 上运行的 Scheduler 指导代理（通过 Windows 54345 端口或 UNIX 上的动态端口）初始化场景会话；控制器（wlrun.exe）在发送请求时发送一份场景的拷贝。

8）代理是由每一个负载发生器上的 Remote Agent Dispatcher 进程（8.0 叫 Remote Command Launcher（RCL））启动的。

9）每个代理根据场景（.lrs）定义文件来决定哪个虚拟用户组和脚本需要在主机上运行，这就是说控制器可以从 DOS 批处理文件（.batch）中启动。

10）控制器通过使用 Windows 操作系统根目录文件夹里的参数值来启动，因为 LoadRunner 被设计成在一个机器上并且一次只能运行一个控制器实例，所以需要使用 Windows 文件夹。

为了在几个应用之间快速的切换，Controller 工作之后会保存在 LoadRunner 的 ini 文件中，然后使用记事本来制作一个批处理文件，在执行 wlrun 之前拷贝应用程序的指定版本的 ini 文件。

11） 在 Vuser 中定义的每个虚拟用户进行的操作都是 LoadRunner 的 VuGen.exe 生成的，当这个程序启动后，它在 Windows 文件夹下存储了 comparamui.ini 文件来保存[LastTablesUsed]下文件的历史，而[ParamDialogDates]项是由“插入”→“新参数”→“数据”来指定。

12）在运行期间，执行结果存储在一个结果文件夹中。

在结果中设置“为每一个设定执行自动创建结果目录”，这样 LoadRunner 会在每次启动一个场景之后自动产生一个递增的结果名。例如，结果名称 Res1 会自动增长到 Res12 或是 Res11-1，错误被写到 Microsoft Access 数据库文件 output.mdb 中。

13）在每一个结果文件夹中，程序自动创建一个 Log 文件夹，在这个文件夹中包含每个组的日志文件，运行结束之后，在 Controller 中查看日志文件，点击 按钮然后在组中点击右键，选择“查看 Vuser 日志”。

14）场景运行的时候，监视器在本地维护每个主机的计数器。

15）场景运行结束后，进程处理.eve 和.lrr 结果文件并且在结果文件夹下创建一个临时的.mdb（M）数据库。在处理大数据量的结果时，为了防止错误发生，通常使用（Microsoft Access）数据库文件。

16）分析模块（8,320K analysisu.exe）使用.mdb 数据库中的数据来产生分析图表和报告。

17）每次设定运行后的 LoadRunner 结果文件 result_name.lrr（也称为分析器文档文件），由分析程序来读取并显示百分位图表。

18）默认情况下，LRReport 文件夹被创建在本地分析机器的 My Documents 文件夹下用来存储分析会话文件。

19）可以使用 HTML 格式产生报告。

20）结果文件格式是由.tem 模板文件控制的。

21）Mercury 的 LR7.8 版中的 Remote Performance Monitoring（RPM）MS-IIS/ASP Web Server 可以安装在 Windows 2000 Server 上（但在 Windows Server 2003 不行）。

负载测试的结果可以使用 Web 浏览器来浏览了。

以上就是 LoadRunnerr 测试的全部过程。

2.5 LoadRunner 测试步骤

LoadRunner 虽然只是一个性能测试工具，但使用它进行性能测试时也有其自身的一个流程。它由方案与计划阶段、创建 Virtual Users、设计测试场景、执行场景和分析结果五个步骤组成。具体的工作流程如图 2-4 所示。

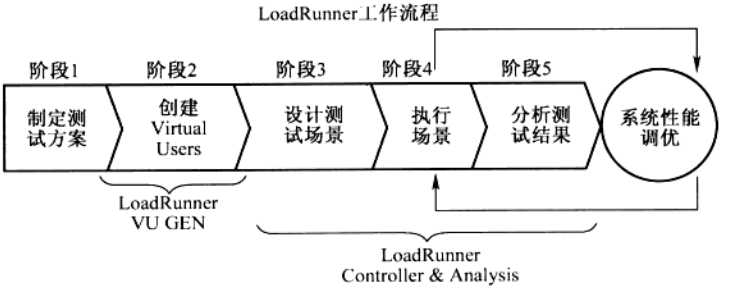


图 2-4 LoadRunner 性能测试过程

图 2-4 中多出系统性能调优的过程，因为进行性能测试的目的是找到系统性能的瓶颈，进而帮助开发工程师对系统性能进行调优，如果不能给开发工程师提出性能调优的有效建议，那么性能测试是做得不够成功的。实际上项目进行测试过程也是这样，性能调优是一个循环的过程，有可能要经历多次测试才能达到目标能力。

下面对这五个阶段进行详细的进述：

1. 方案与计划阶段

任何一种测试类型，不管是性能测试还是功能测试，这个步骤都是必不可少的。在方案与计划中要定义性能测试的功能模块、期望达到的性能目标、测试时间的安排、业务模型和场景模型。业务模型和场景模型是最重要的部分之一，业务模型一定要找业务专家来协助完成，如果业务模型设计出错，那么测试出来的数据是不可靠的，这样性能测试就已经失败了；场景模型的设计是模拟大量用户进行并发操作的一个过程，是为了让脚本运行更接近用户真实的使用情况。时间安排、期望得到的协助资源也很重要，这里为什么要将期望得到的协助资源列出来，是因为在现实项目进行的过程中，仅仅靠一个性能测试工程师的能力来解决所有问题是不现实的，在整个性能测试过程中必须要借助他人的帮助才能完成，一个人的能力再强也是无法实现的。最后一部分是风险分析，即性能测试前要对可能存在的风险进行分析，这样就可以在性能测试前期就做好准备，避免在测试过程

中去补救。只有做好了这些，性能测试才能顺利的进行，才能做到事半功倍。

2. 创建 Virtual Users 阶段

在进行脚本录制之前，首先需要和业务专家进行沟通，因为业务专家才最了解业务和用户使用，只有在录制时了解尽可能多的业务知识才能更好地录制脚本，这样业务模型才能做到最好。熟悉业务流程之后，接下来就是进行脚本录制，使用 LoadRunner 进行脚本录制的过程很简单，点击“开始录制”按钮，然后进行操作，直到结束，点击“停止”按钮，脚本即录制完成。录制完脚本之后必须对脚本进行编辑，在脚本编辑的过程中要注意以下几个方面：

- 定义集合点
- 定义事务
- 参数化
- 关联

定义集合点是为了让虚拟用户进行真正意义上的并发操作，如果只设置虚拟用户数，但没有设置集合点，这样就无法保证所有的虚拟用户真的在同一时刻进行并发操作，关于集合点的具体使用在后面会详细讲到。

定义事务是为了更好地去度量业务流程所消耗的时间。

参数化：参数化是脚本编辑最重要的技巧之一，为了使脚本满足测试的需求，减少脚本的重复程度就必须进行参数化，参数化的数据来源有两种：一些历史的数据和测试过程中构建的虚拟数据。

关联：关联也是编辑脚本最重要的技巧之一，当从服务器返回来的值是一个动态的、变化的值，像 cookie 和 session 等信息，这样在录制和回放时，从服务器端读取的数据也就是不同的，回放脚本就会出错，这时就得通过关联技术来解决脚本回放的问题。

3. 设计测试场景

测试场景用来描述测试活动中发生的各种事件。一个场景包括运行 Vuser 活动的负载机列表、测试脚本列表、大量的 Vuser 和 Vuser 组。

场景设计是通过 LoadRunner 的控制台来控制的。

当然这里还包含了场景建模技术，就是如何才能设计一个场景来尽可能地接近用户真实使用的场景，包括集合点应用、IP 欺骗技术、并发用户数、各脚本用户并发数等，这些问题都应该在场景设置的过程中予以考虑。

4. 执行场景

场景设计完成后，开始执行场景，LoadRunner 内含集成的实时监测器，在负载测试过程中可以观察系统的运行情况，以及对操作系统、数据库、中间插件等进行实时监控。这样就可以更好地去分析系统运行时性能的指标，更快地去发现问题。

5. 分析结果

当测试结束之后，LoadRunner 会生成一个结果文件，这个结果文件会收集所有的测试数据，并且自带了一些高级分析工具和报告工具。

常用的结果分析技术有：合并图表、关联图表、页面细分和钻取技术。

3

Vuser 发生器

Vuser 发生器（Visual User Generator，简称为 VuGen）用来捕获最终用户业务流程和创建自动化测试脚本，即生成测试脚本。VuGen 是录制测试脚本、编辑与完善测试脚本的一个平台，支持 C 语言语法。

本章主要包括以下内容：

- 脚本录制
- Recording Options 设置
- Run-Time Settings 设置
- 脚本完善

3.1 脚本录制

启动 Visual User Generator，创建一个新的脚本，开始录制脚本，在录制脚本过程中，VuGen 会自动捕获操作过程中客户端与服务器端进行通信的所有数据。这里涉及的关键点是如何选择录制协议。

脚本开发主要包括四大步骤：计划、录制脚本、脚本增强和单机调试脚本，如图 3-1 所示。

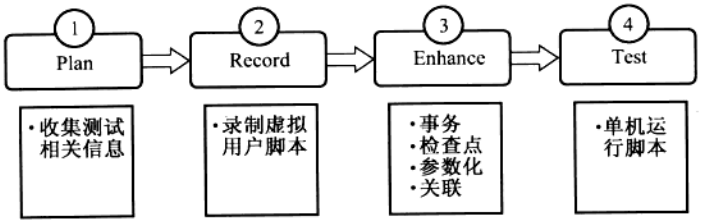


图 3-1　脚本开发过程

3.1.1 如何选择协议

在创建一个新的脚本时，首先会弹出一个对话框，在该对话框中选择录制时需要的协议，这步非常重要，选择的协议将直接影响到录制后的脚本是否理想，如何选择录制协议是录制前必须要解决的问题。

各种协议和相关头文件的对应关系如图 3-2 所示。

协议	头文件
AJAX(Click&Script)	web_ajax.h
Citrix	ctrxfuncs.h
COM/DCOM	lrc.h
Database	lrd.h
FTP	mic_ftp.h
General C function	lrun.h
IMAP	mic_imap.h
LDAP	mic_midap.h
MAPI	mic_mapi.h
Oracle NCA	orafuncs.h
POP3	mic_pop3.h
ROP	lrrdp.h
SAPGUI	as_sapgui.h
SAP(Click&Script)	sap_api.h
Siebel	lrdsiebel.h
SMTP	mic_smtp.h
Terminal Emulator	lrrte.h
WAP	as_wap.h
Web(HTML\HTTP)	as_wab.h
Web(Click&Script)	web_api.h
Web Services	wssoap.h
Windows Sockets	lrs.h

图 3-2 各协议和相关头文件对应关系

选择协议的常用方法主要有以下几种：

1）最简单的方法就是向开发工程师确认数据通信所采用的协议，因为开发工程师最清楚应用程序采用的是何种通信协议。

2）没有开发工程师支持时，可以通过概要或详细设计手册获知所使用的协议。

3）使用协议分析工具捕获通信时的数据包并进行分析，然后确定被测对象所使用的协议。在使用协议分析工具分析协议过程中一定要摒除底层协议，不要被底层协议所迷惑。

4）根据以往测试经验来判断被测试对象采用的协议，这种方法具有猜测性，有时候不一定准确。LoadRunner 提供了多种协议，具体的协议分类见表 3-1。

表 3-1 协议分类表

应用类型		建议选用协议
Web 网站（J2EE、.NET）		Web（HTTP/HTML）
FTP 服务器		File Transfer Protocol（FTP）
邮件服务器		Internet Messaging Application Protocol（IMAP）
		Post Office Protocol（POP3）
		Simple Mail Transport Protocol（SMTP）
C/S	客户端以 ADO、OLEDB 方式连接后台数据库	Microsoft SQL Server Oracle、Sybase、DB2、Informix
	以 ODBC 方式连接后台数据库	ODBC
	没有后台数据库	Socket
分布式组件		COM/DCOM、EJB
无线应用		WAP、Palm

创建新脚本时，会弹出一个对话框，LoadRunner 提供三种选择协议的方式：单协议脚本、多协议脚本和最近使用过的协议。

1）单协议脚本：创建单协议 Vuser 脚本，在对话框中选择录制时需要的协议，如图 3-3 所示。

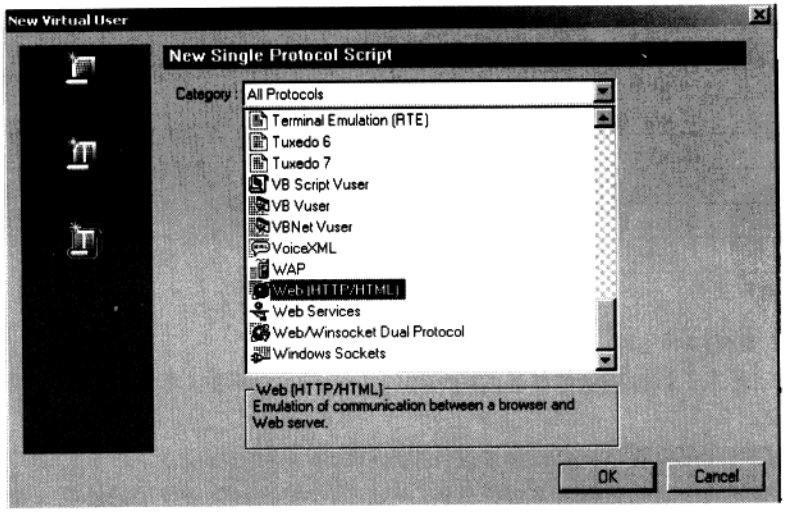


图 3-3 单协议脚本

2）多协议脚本：创建多协议 Vuser 脚本。在 Available Protocols 中选择一个或多个协议，点击右箭头，将其移入到 Selected Protocols 部分中，同样，在 Selected Protocols 中选择一个或多个协议，点击左箭头可以移除选中的协议，如图 3-4 所示。

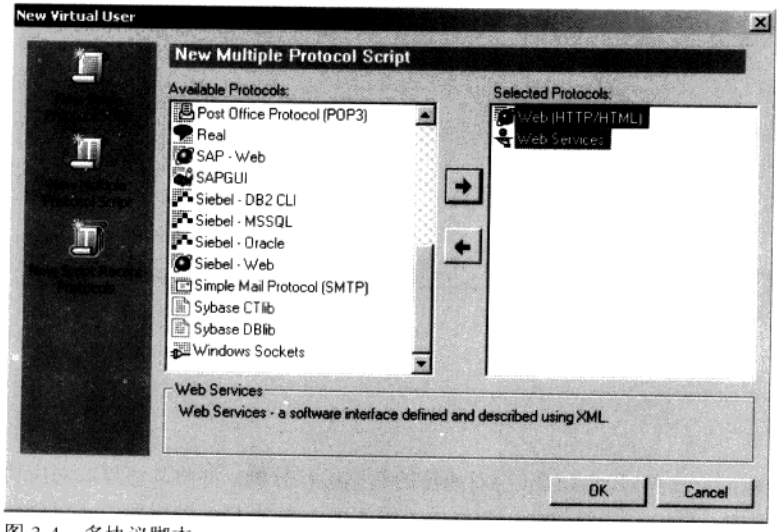


图 3-4 多协议脚本

3）最近使用过的协议：从最近录制脚本的协议列表中，选择一种协议进行录制，如图 3-5 所示。

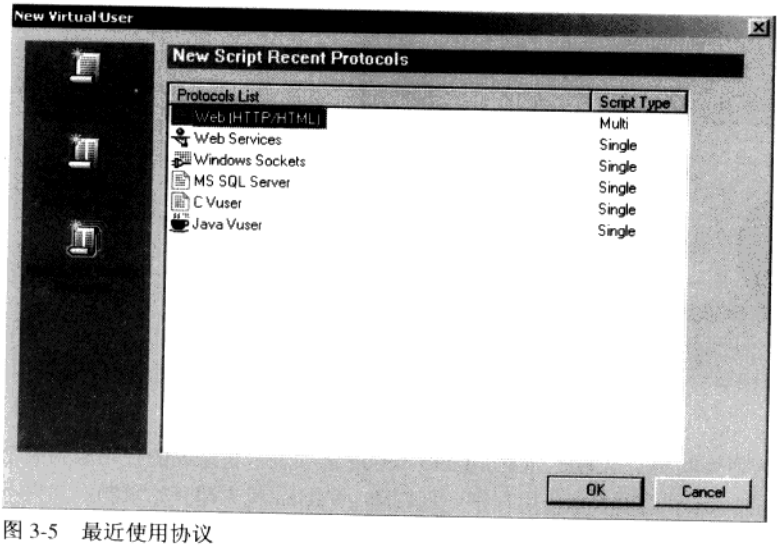


图 3-5 最近使用协议

3.1.2 开始录制脚本

协议选择好后可以开始录制脚本。这里以 Web（HTTP/HTML）协议为例进行录制。

VuGen 录制浏览器主要是通过代理的方式来实现的。开始录制时，VuGen 打开浏览器，并以 VuGen 作为代理来访问目标服务器。这样，VuGen 就可以捕获客户端与服务器之间通过的数据包，如图 3-6 所示。

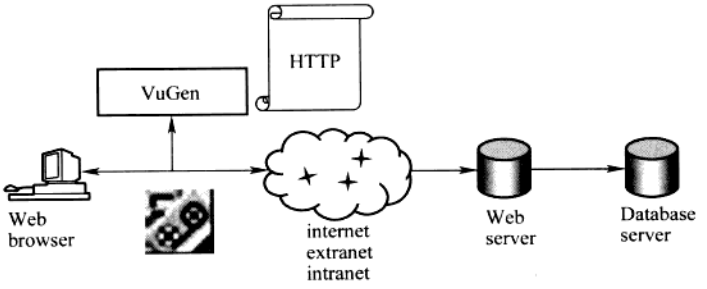


图 3-6 VuGen 录制原理图

在使用 VuGen 进行录制用户操作时，VuGen 会对捕获的数据进行分析，并将其还原成对应协议的由 API 组成的脚本。同时，VuGen 会将这些函数生成的脚本插入到 VuGen 编辑器中，以创建原始的 Vuser 脚本。

录制时系统弹出一个录制窗口，如图 3-7 所示。

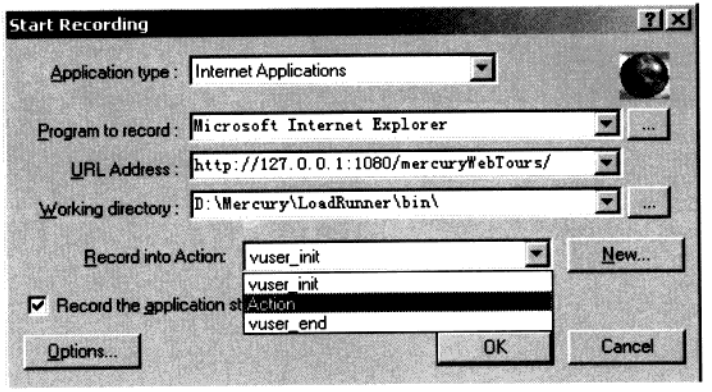


图 3-7 录制窗口

在 URL Address 中输入要录制的站点地址。Record into Action 选项表示将录制的代码放到哪个部分。Record the application startup 选项表示应用程序一旦启动，VuGen 就立即开始录制；如果不选中，应用程序启动后，VuGen 会弹出如图 3-8 所示的对话框，并且暂时不会进行录制，当用户操

作应用程序到需要录制的地方时，点击 Record 按钮，VuGen 才开始录制。默认情况下 Record the application startup 是选中的状态。点击 Record 按钮开启录制。

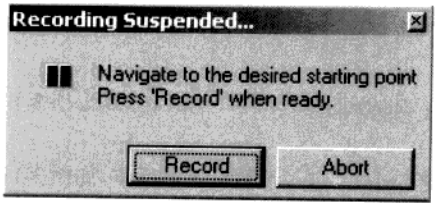


图 3-8 暂停录制

开始录制后，会出现如图 3-9 所示的工具栏。

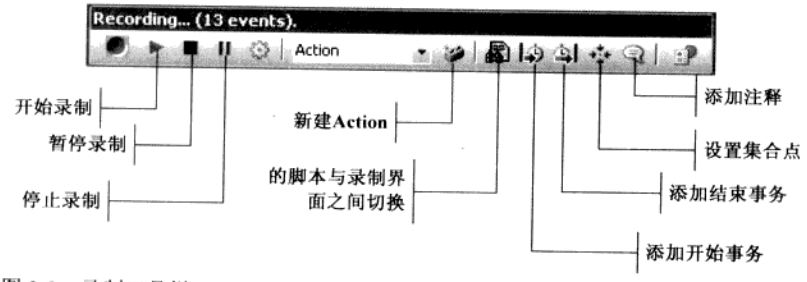


图 3-9 录制工具栏

该工具条从左到右依次代表开始录制、暂停录制、停止录制、新建 Action、在脚本与录制界面之间切换、添加开始事务标识、添加结束事务标识、设置集合点和添加注释。

录制过程中，LoadRunner 会自动记录用户的操作。录制完成后，点击“停止录制”按钮结束录制，这时 VuGen 会自动生成一个脚本，如图 3-10 所示。

```
Action()
{

    lr_think_time(9);


    web_submit_data("index.php",
    web_find("web_find",
    web_url("index.php_2",

    web_url("projectedit.php",

    return 0;
}
```

图 3-10 脚本录制

3.2 Recording Options 设置

在进行录制时，首先要对录制的一些参数进行设置，只有将这些参数设置好，才能录制并生成需要的脚本。

首先是 Recording Options 设置，需要注意的设置项有：Recording 选项卡、Advanced 选项卡和 Correlation 选项卡。

在 Tools 菜单中选择 Recording Options 或直接按快捷键 Ctrl+F7 进入参数设置对话框，如图 3-11 所示。

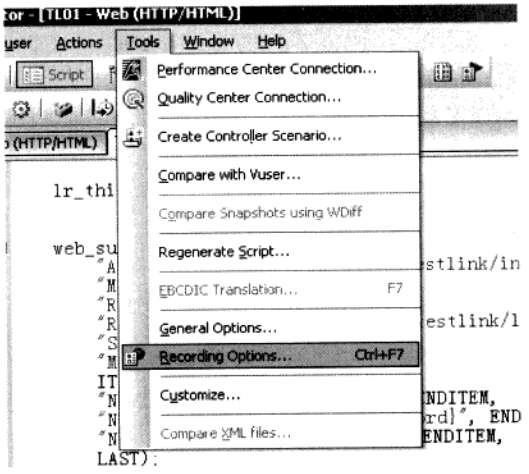


图 3-11 Recording Options 设置

3.2.1 Recording 选项卡

在 Recording Options 对话框中，选择 Recording 选项卡。Recording Level 包含两种录制模式：HTML-based script 和 URL-based script，如图 3-12 所示，默认情况下选中 HTML-based script 录制方式。当然，两种录制模式也存在差别。

HTML-based script 方式：对每个页面录制形成一条语句，对 LoadRunner 来说，在该模式下，访问一个页面，首先会与服务器建立连接并获取页面的内容，然后从页面中分解得到其他的元素（component），最后建立几个连接分别获取相应的元素。

URL-based script 方式：将每条客户端发出的请求录制成一条语句，对 LoadRunner 来说，在该模式下，一条语句只能建立一个到服务器的连接，并将通信过程中的很多隐藏的信息都录制出来（如 session、cookie）。LoadRunner 提供了 web_concurrent_start()和 web_concurrent_end()函数模拟 URL-based script 的工作方式。

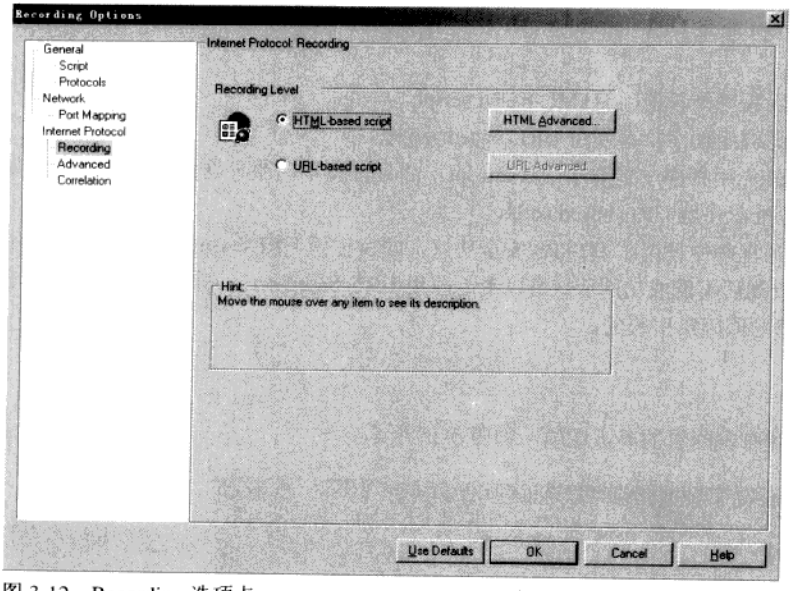


图 3-12 Recording 选项卡

图 3-13 和图 3-14 是分别使用这两种录制模式录制同一个系统登录过程的脚本。

```
web_submit_form("login.pl",
    "Snapshot=t2.inf",
    ITEMDATA,
    "Name=username", "Value=arivn", ENDITEM,
    "Name=password", "Value=02321408", ENDITEM,
    "Name=login.x", "Value=0", ENDITEM,
    "Name=login.y", "Value=0", ENDITEM,
    LAST);

web_image("SignOff Button",
```

图 3-13 HTML 方式录制

```
lr_think_time(9);

web_submit_data("login.pl",
    "Action=http://127.0.0.1:1080/mercuryWebTours/login.pl",
    "Method=POST",
    "RecContentType=text/html",
    "Referer=http://127.0.0.1:1080/mercuryWebTours/nav.pl?in=home",
    "Snapshot=t9.inf",
    "Mode=HTTP",
    ITEMDATA,
    "Name=userSession", "Value=100280.377480761fVfiVHVpQVzzzzzHDAQtQpADVf", ENDITEM,
    "Name=username", "Value=arivn", ENDITEM,
    "Name=password", "Value=02321408", ENDITEM,
    "Name=JSFormSubmit", "Value=off", ENDITEM,
    "Name=login.x", "Value=36", ENDITEM,
    "Name=login.y", "Value=12", ENDITEM,
    LAST);

web_concurrent_start(NULL);
```

图 3-14 URL 方式录制

选择 HTML-based script 还是 URL-based script，应该根据实际需要进行，下面是一些常见的参考原则：

1）基于浏览器的应用程序推荐使用 HTML-based script。

2）不是基于浏览器的应用程序推荐使用 URL-based script。

3）如果基于浏览器的应用程序中包含了 JavaScript，并且该脚本向服务器发送了请求，比如 DataGrid 的分页按钮等，推荐使用 URL-based script。

4）基于浏览器的应用程序中使用了 HTTPS 安全协议，建议使用 URL-based script。如果使用 HTML-based script 模式录制后不能成功回放，可以考虑改用 URL-based script 模式来录制。因为这种情况多是由上面所列举的原因所引起的。

3.2.2 Advanced 选项卡

Advanced 选项是设置脚本回放的高级选项，如图 3-15 所示。

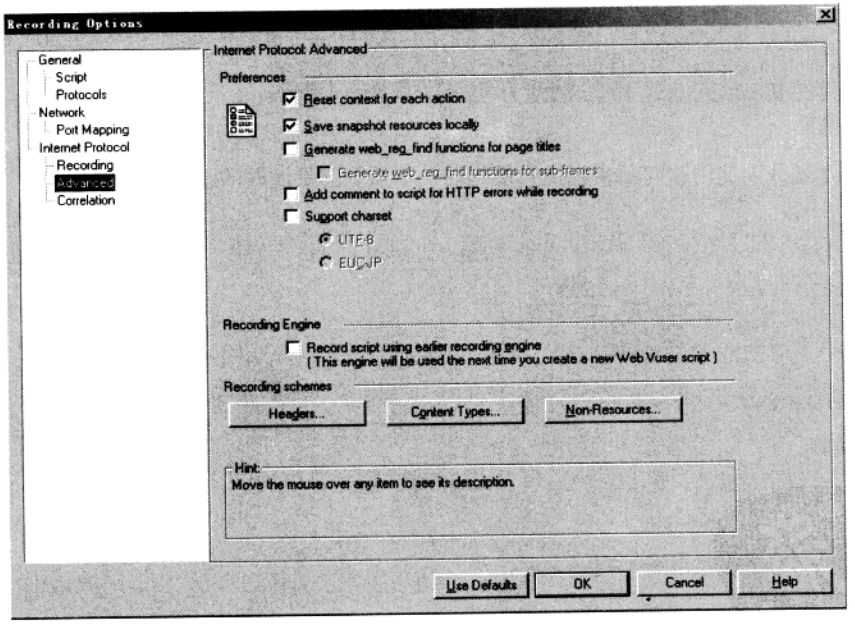


图 3-15 Advanced 选项卡

- Save snapshot resources locally：表示运行结果中保存一个快照。
- Add comments to script for HTTP errors while recording 表示出现错误时会自动添加注释。

点击 Headers…按钮，会弹出 HTTP Headers 配置对话框，如图 3-16 所示。在该对话框中可以选择需要录制的 Headers，以便服务器能够正确处理编辑信息。需要注意的是 Accept-Language 选项，像 Websphere 这类服务器会根据 HTTP 请求中的 Header 来确定编码。

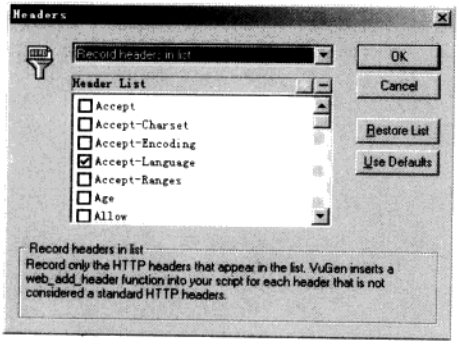


图 3-16 添加 HTTP Headers

3.2.3 Correlation 选项卡

Correlation 选项卡用来对脚本中的关联属性进行设置，如图 3-17 所示。选中需要的关联规则，在录制脚本时会自动进行关联。如果当前的这些关联规则无法满足录制的需求，那么可以点击 New Application 按钮来新建一个关联，再点击 New Rule 按钮为该关联新建一个规则。

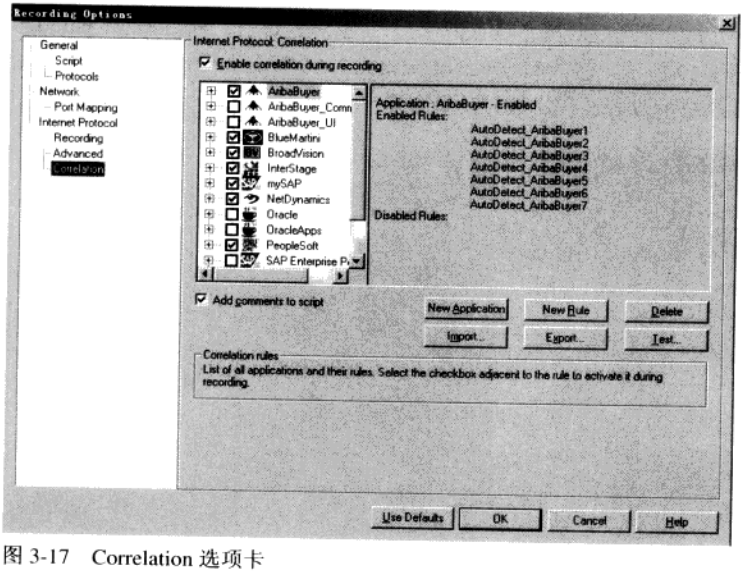


图 3-17 Correlation 选项卡

3.3 Run-Time Settings 设置

Run-Time Settings 设置主要是用来设置脚本回放过程的一些参数。在菜单 Vuser 中选择

Run-Time Settings 或按快捷键 F4，如图 3-18 所示。需要注意的设置项有：Run Logic 选项卡、Pacing 选项卡、Think Time 选项卡和 Miscellaneous 选项卡。

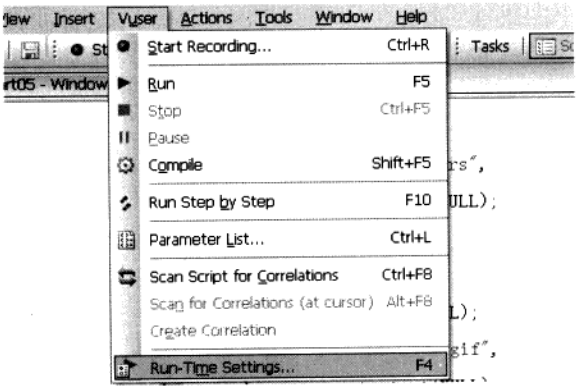


图 3-18 Run-Time Settings

3.3.1 Run Logic 选项卡

Run Logic 选项卡主要是用来设置运行时脚本迭代的次数，可以通过更改 Number of iterations 的值来设置迭代的次数，如图 3-19 所示，修改 Number of Iterations 的值只对 Run 部分的脚本迭代次数有影响，而对 Init 和 End 部分的脚本迭代次数并没有影响。

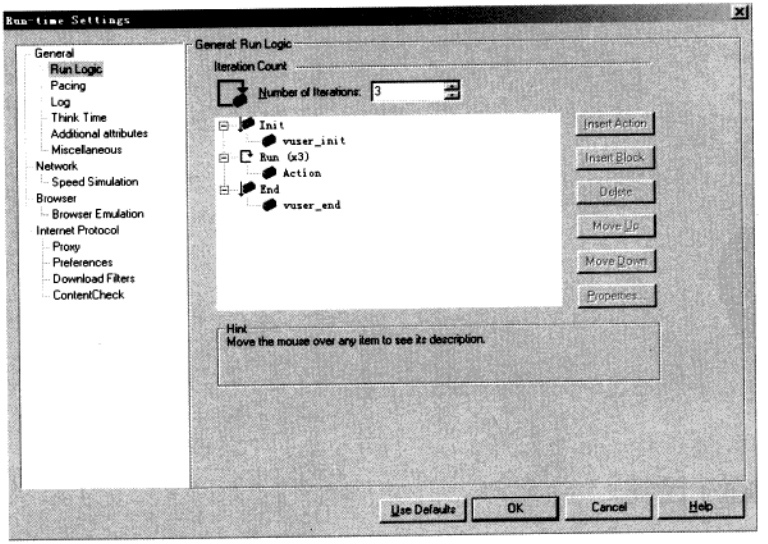


图 3-19 Run Logic 选项卡

3.3.2 Pacing 选项卡

Pacing 选项卡主要用来设置脚本迭代过程中脚本之间的时间间隔，如在第 N 次脚本迭代完成后，等待 5 秒钟后进行第 N+1 次脚本迭代，如图 3-20 所示。

- As soon as the previous iteration ends：在多次迭代时，上一次迭代执行结束后马上执行下一次迭代。
- After the previous iteration ends：可以设置为 Fixed 或 Random 方式。Fixed 方式表示上一次迭代执行结束后，等待一个固定时间后，再执行下一次迭代；Random 方式表示上一次迭代执行结束后，等待一个随机时间后，再执行下一次迭代，随机时间范围为设置的范围。
- At Fixed/Random intervals：表示无论上一次迭代执行是否完成，到达规定的时间就开始执行下一次迭代，包含 Fixed 和 Random 两种方式。Fixed 表示一个固定的时间长度；Random 表示一个随机的时间长度，随机值范围为设置的范围。

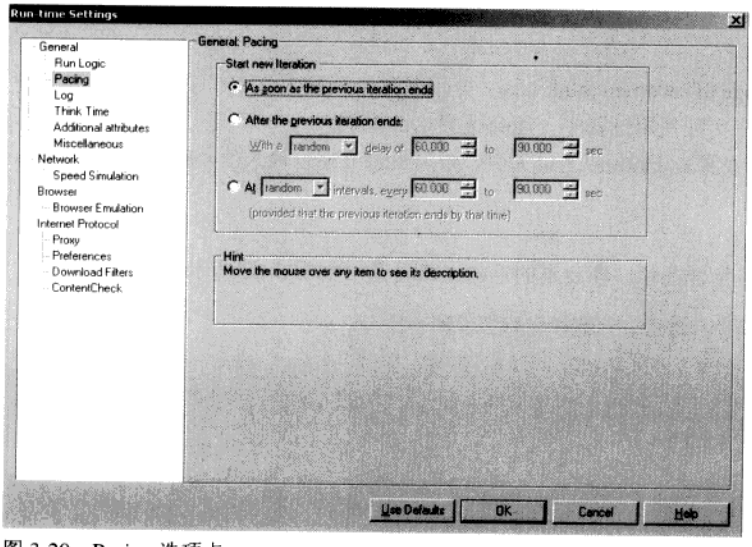


图 3-20 Pacing 选项卡

3.3.3 Think Time 选项卡

Think Time 选项卡用来设置用户操作的思考时间，如图 3-21 所示。

- Ignore think time：运行脚本时忽略思考停顿时间。
- Replay think time：设置思考时间的延迟。
- As recorded：根据录制的思考时间来运行。
- Multiply recorded think time by：根据录制时思考时间的整数倍来运行。

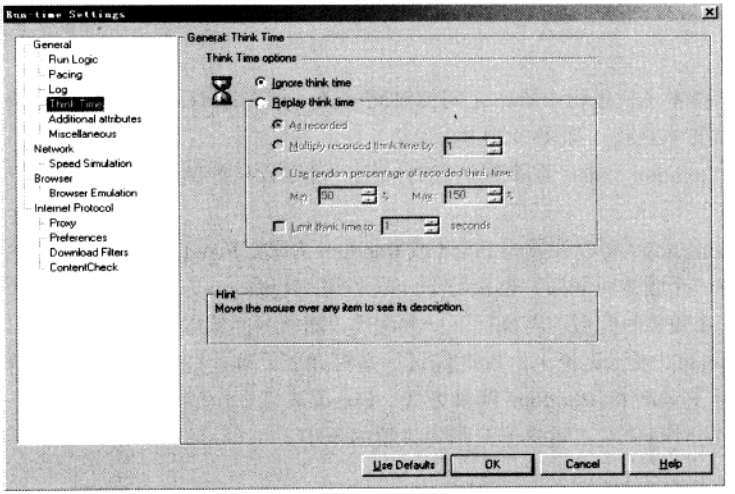


图 3-21 Think Time 选项卡

- Use random percentage of recorded think time：分别设置一个最大值和一个最小值，并从中选出一个随机值。在实际使用过程中一般会选择这种模式。
- Limit think time to：设置 think time 的最大值。如果录制值超过最大值，就以最大值为准。

3.3.4 Miscellaneous 选项卡

Miscellaneous 选项是一个复合选项，涉及到的功能比较复杂，如图 3-22 所示。

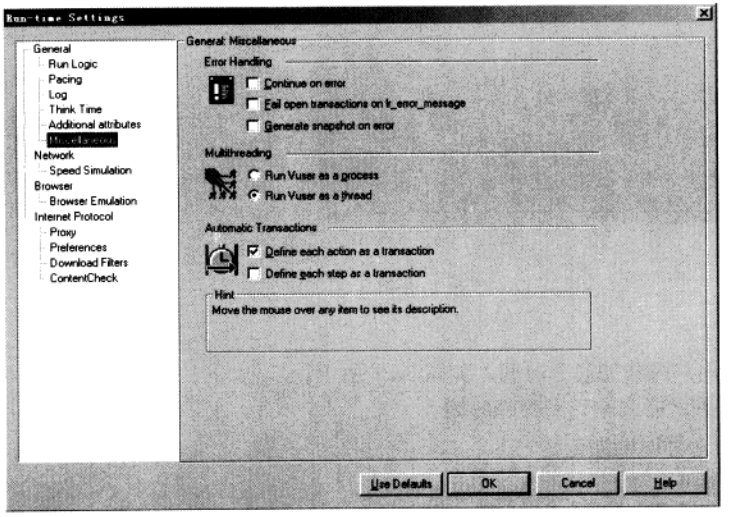


图 3-22 Miscellaneous 选项卡

这里包括三个设置项：Error Handling、Multithreading 和 Automatic Transactions。各项设置的含义如下：

Error Handling 选项：表示脚本运行出现错误时所采取的措施，默认使用缺省值。

Multithreading 选项：表示运行时把虚拟用户当作进程还是线程来处理。Run Vuser as a process 表示把虚拟用户当作进程来处理；Run Vuser as a thread 表示把虚拟用户当作线程来处理。一般会选择把虚拟用户当作进程来处理，因为系统上线后，用户的操作都是以进程的方式进行。

Automatic Transactions 选项：设置事务的模式。Define each action as a transaction：将一个 action 看作一个事务；Define each step as a transaction：将每一个操作步骤看作一个事务。

3.4 脚本完善

脚本录制完成并不代表这些脚本很接近用户的真实使用情况，为了使这些脚本更接近用户的真实使用情况，需要对脚本进行完善。

首先，为了衡量服务器对某个动作处理的响应时间，需要定义事务（Transaction）。例如在脚本中有一个登录动作，为了衡量多用户同时执行登录服务器的响应时间，可以将此操作定义为一个事务。LoadRunner 运行到事务开始点时即开始计时，直到运行至该事务结束点结束计时。这个事务的运行时间在结果中会反映出来，LoadRunner 对插入事务的个数没有限制。

其次，为了衡量加重负载的情况下服务器的性能情况，需要插入集合点。例如为了模拟 30 个用户同时登录的情况，做到真正的并发，就必须添加一个集合点，用来衡量服务器的性能。

最后，在录制脚本的过程中添加注释，因为录制完成之后都是代码，在操作一些复杂的业务时，可能很难一下找到相应的位置进行像参数化一类的动作，这时，注释显得很重要。同时添加注释也可以增强脚本的可读性，方便其他的工程师阅读。

3.4.1 插入事务

插入事务有两种方法：一种是在脚本录制过程中插入开始和结束事务点；另一种是在编辑脚本时插入开始和结束事务点。一般情况下选择在脚本录制过程中插入开始和结束事务点。这样做有两个优点：第一，保证不会遗漏需要插入的事务点；第二，避免在脚本录制完成后找不到确切的插入事务点的地方，对于有经验的性能测试工程师这可能不是问题，但是对于一个新手很有可能无法确切地找到插入事务点的位置。

方法一：在脚本录制过程中插入开始和结束事务点

当脚本录制到需要插入事务点时，在录制工具栏中点击“插入开始事务点”按钮，如图 3-23 所示。在弹出的“开始事务”对话框中输入开始事务名称，如图 3-24 所示。事务点名称的格式应该保持统一，使其具有代表性，遵守脚本编辑命名规则。

图 3-23 录制中插入开始事务点

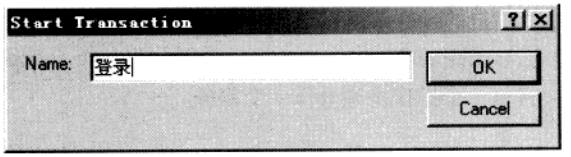


图 3-24 输入开始事务名称

事务的开始点与事务的结束点具有一一对应的特点，即在脚本中插入了一个事务开始点后一定要在接下来的过程中插入事务结束点，插入结束事务点与插入开始事务点的操作一致，当该业务流程执行完毕后需要添加结束事务点，在录制工具栏中点击“插入结束事务点”按钮，如图 3-25 所示。在弹出的“结束事务”对话框中输入结束事务名称，一般在下拉框中选择，因为下拉框中包含了所有开始事务点的名称，如图 3-26 所示。

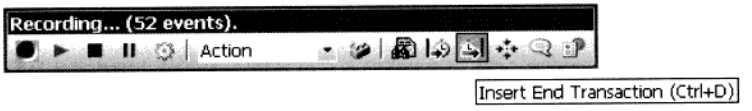


图 3-25 录制中插入结束事务点

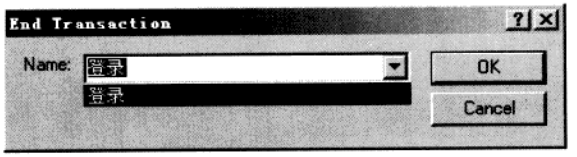


图 3-26 输入结束事务名称

方法二：在脚本录制完成后插入开始和结束事务点

在生成的脚本中找到要插入开始事务点的地方，选择菜单 Insert→Start Transaction（Ctrl+T），如图 3-27 所示。在弹出的“开始事务”对话框中输入开始事务的名称，如图 3-28 所示。

插入结束事务点的过程与插入开始事务点的步骤是一致的，在生成的脚本中找到要插入结束事务点的地方，选择菜单 Insert→End Transaction（Ctrl+T），如图 3-29 所示。在弹出的“结束事务”对话框中输入结束事务的名称，如图 3-30 所示。

在“结束事务”对话框中，比在脚本录制过程中插入结束事务点时的“结束事务”对话框中多出一项 Transaction Status，其事务状态有四种可选：LR_AUTO、LR_PASS、LR_FAIL 和 LR_STOP。

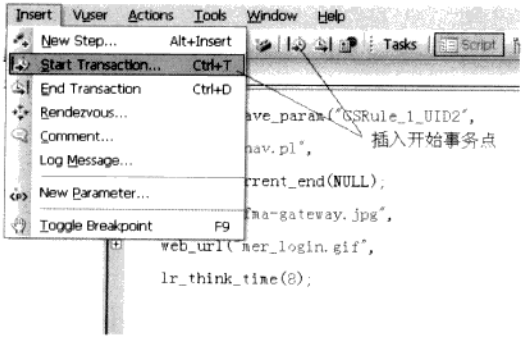


图 3-27 脚本中插入开始事务点

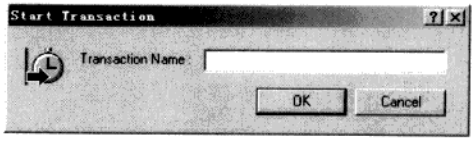


图 3-28 输入开始事务名称

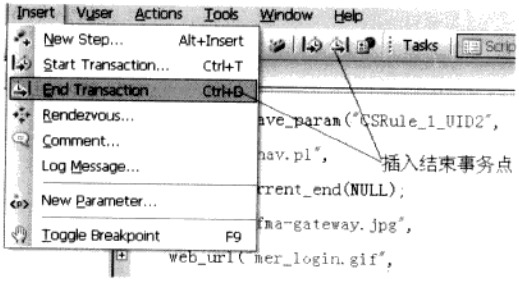


图 3-29 脚本中插入结束事务点

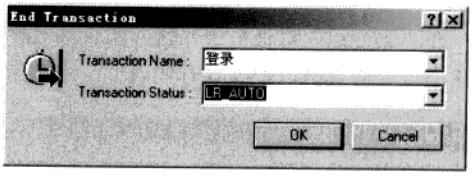


图 3-30 输入结束事务名称

- LR_AUTO：事务的状态被自动设置，如果事务执行成功，状态设置为 PASS；如果事务执行失败，状态设置为 FAIL；如果事务异常中断，状态设置成 STOP。
- LR_PASS：事务执行成功，代码返回的状态是 PASS。
- LR_FAIL：事务执行失败，代码返回的状态是 FAIL。
- LR_STOP：事务异常中断，代码返回的状态是 STOP。

3.4.2 插入集合点

集合点可以在脚本录制过程中插入，也可以在脚本录制完成后插入，但需要注意的是，集合点只能插入 Action 部分的脚本中，不能插入 vuser_int 和 vuser_end 两部分脚本中。

方法一：在脚本录制过程中插入集合点

在脚本录制过程中，当录制到需要插入集合点的位置时，点击录制工具栏中“插入集合点”按钮，如图 3-31 所示。

方法二：在脚本录制完成后插入集合点

在脚本录制完成后，找到脚本中需要插入集合点的位置，选择菜单 Insert→Rendezvous，如图 3-32 所示。

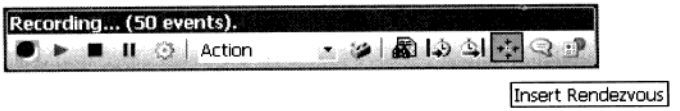


图 3-31 录制中插入集合点

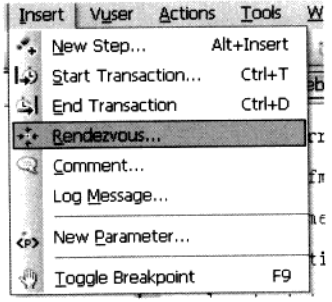


图 3-32 在脚本中插入集合点

在“集合点”对话框中，输入集合点名称，如图 3-33 所示，和定义事务点名称一样，集合点名称应该保持统一，使其具有代表性，遵守脚本编辑命名规则。

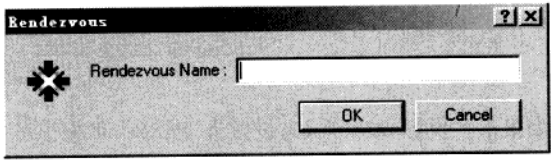


图 3-33 输入集合点名称

3.4.3 插入注释

注释可以在录制脚本过程中插入，也可以在脚本录制完成后插入。

方法一：脚本录制过程中插入注释

在脚本录制过程中，当需要对录制的脚本进行注释时，点击录制工具栏中“插入注释”按钮，如图 3-34 所示。

方法二：脚本录制完成后插入注释

脚本录制完成后，在脚本需要插入注释的地方，选择菜单 Insert→Comment，如图 3-35 所示。

图 3-34 在录制中插入注释

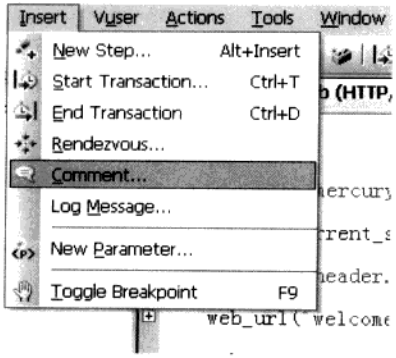


图 3-35 在脚本插入注释

在弹出的“插入注释”对话框中，输入要注释的内容，如图 3-36 所示。

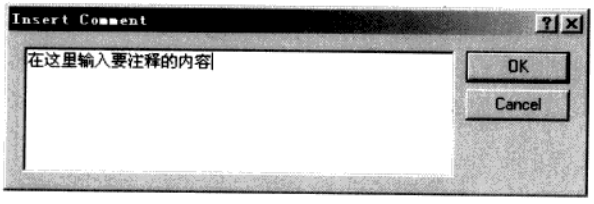


图 3-36 输入注释内容

4

Controller 控制器

Controller 组件是 LoadRunner 的控制中心，主要包括场景设计和场景执行两部分。在 VuGen 中编辑完脚本并将脚本加载到 Controller 组件中，即开始对脚本运行时的场景进行设计，当场景设计完成后，即可执行该场景。场景设计主要是依据需求说明书制定脚本如何执行的策略，使脚本的运行更接近真实用户使用。场景执行是指当场景设计完成后手动运行场景，在场景执行过程中可以实时对场景产生的数据进行监控。

本章主要讲述以下几部分内容：

- 场景类型介绍
- 场景设计
- 场景执行

4.1 场景类型介绍

Controller 控制器提供了手动和面向目标两种测试场景。一般情况下使用手动场景设计方法来设计场景，手动设计场景最大的优点是能够更灵活地按照需求来设计场景模型，使场景能更好地接近用户的真实使用。面向目标场景则是测试性能是否能达到预期的目标，在能力规划和能力验证的测试过程中经常使用到。

Controller 的启动有两种方式，可以在“开始”菜单→“所有程序”→Mercury LoadRunner→Applications→Controller 中启动，如图 4-1 所示。也可以在 VuGen 发生器菜单 Tools→Create Controller Scenario 中启动，如图 4-2 所示。

4.1.1 手动测试场景

启动 Controller 控制器后，会弹出“新建场景”对话框，如图 4-3 所示。

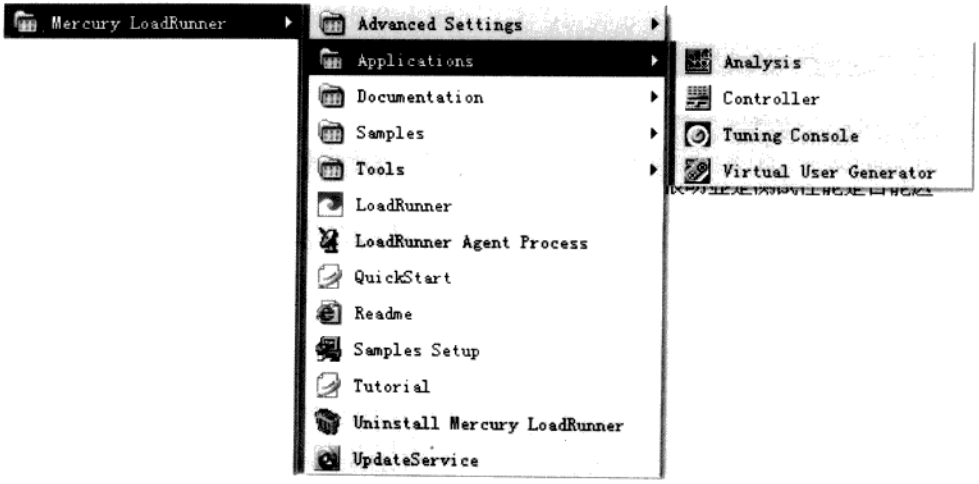


图 4-1 “开始”菜单启动 Controller

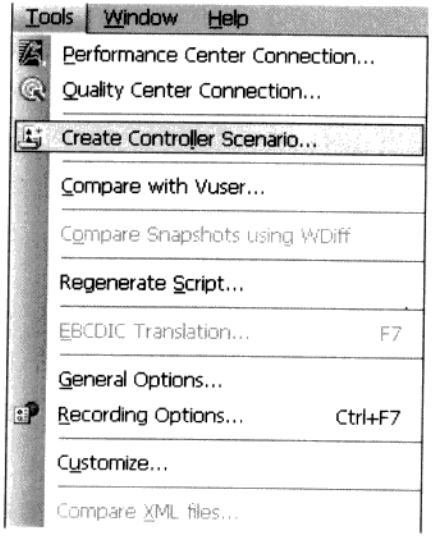


图 4-2 Tools 菜单启动 Controller

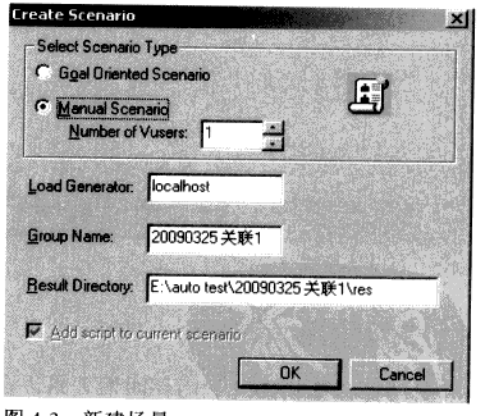


图 4-3 新建场景

“新建场景”对话框中有两种场景方法可以选择：手动场景和面向目标场景。一般手动测试场景使用得较多，因为手动场景更灵活，可以更好地接近用户真实的使用情况。

手动场景又包含两种模式：用户组模式与百分比模式，这两种模式的不同之处在于计算虚拟用户的方式不同。

手动用户组模式如图 4-4 所示。

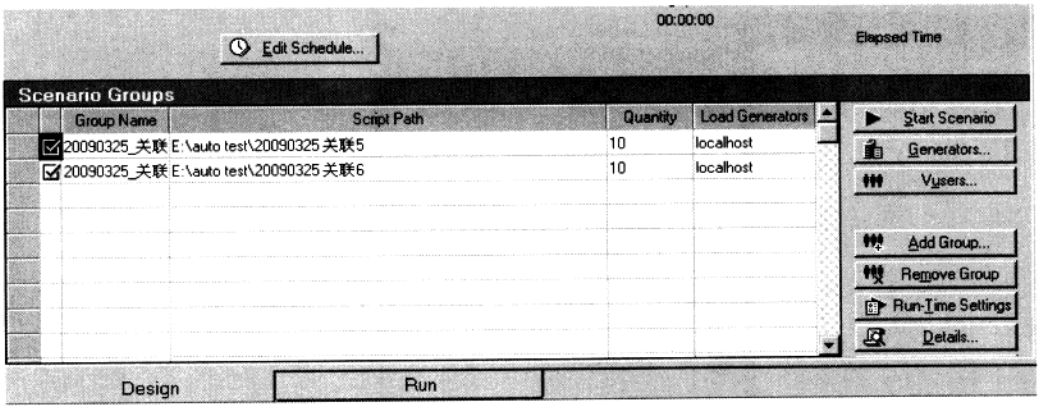


图 4-4 用户组模式

在 Controller 控制器中，点击菜单 Scenario→Convert Scenario to the Percentage Mode 即可切换到百分比模式，如图 4-5 所示。

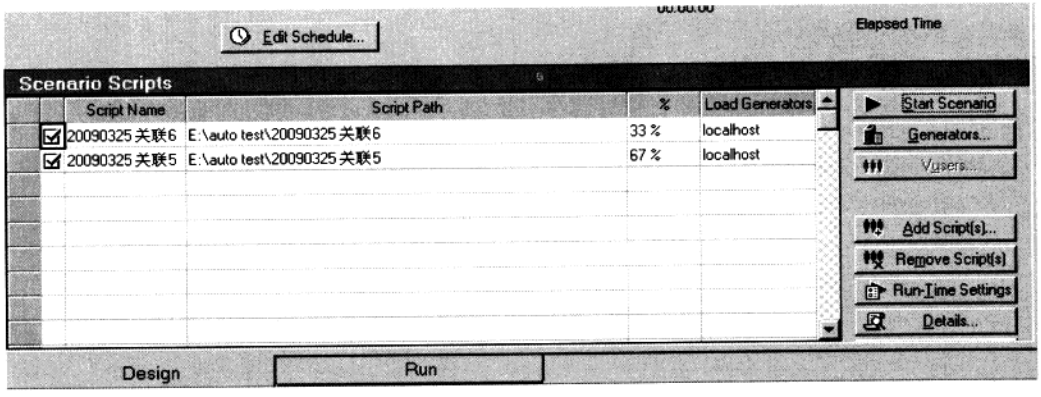


图 4-5 百分比模式

4.1.2 面向目标测试场景

面向目标场景是一个闭环回馈关系。在这种场景模式下，首先定义要达到的目标，接着 LoadRunner 会自动基于该目标创建场景，在场景运行过程中，LoadRunner 会不断地将结果与目标相比较，以决定下一步如何执行。

面向目标测试场景提供了 Virtual Users、Hits per Second、Transactions per Second、Transaction Response Time 和 Pages per Minute 五种目标。

如图 4-6 所示是面向目标测试场景界面。

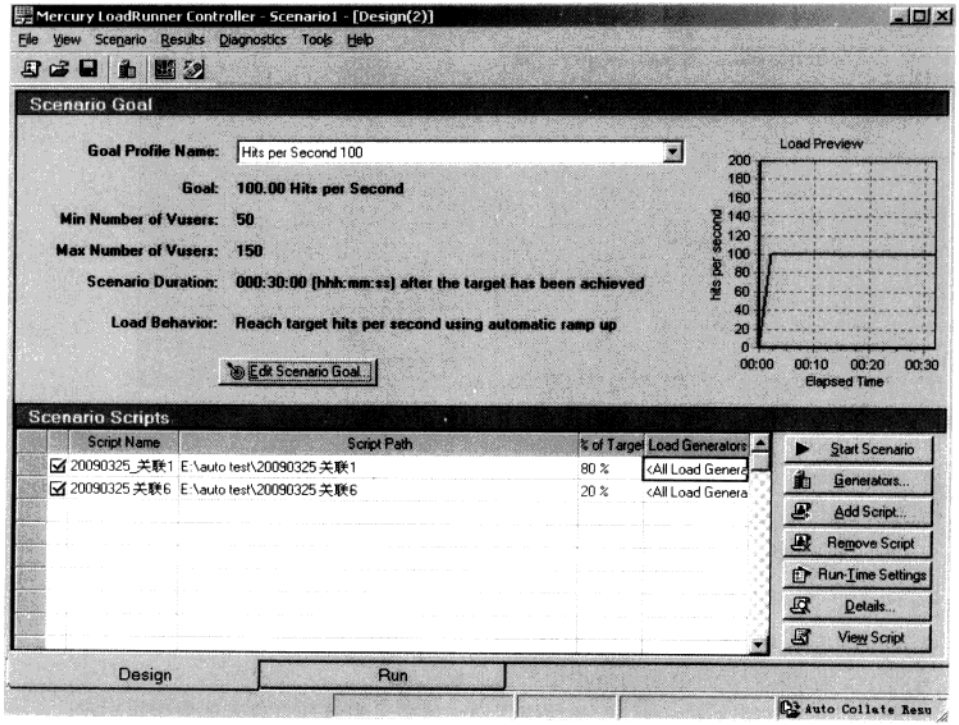


图 4-6 面向目标测试场景

4.2 场景设计

本节主要介绍手动和面向目标两种场景设计，包括 Schedule、View Script 和 Generator 参数的设置。手动和面向目标两种场景的不同主要体现在 Schedule 参数的设置上，二者的 View Script 和 Generator 参数设置都是一致的，故主要介绍手动场景 Schedule 配置和面向目标场景 Schedule 配置两部分。

4.2.1 手动场景 Schedule 配置

在手动场景设计界面，点击 Edit Schedule 按钮，进入 Schedule 配置界面，如图 4-7 所示。

Schedule 配置主要是用来设置用户的行为方式，这里包括按场景计划和按用户组计划两种。

1. 场景名称（Schedule Name）

可以添加一个场景、对场景进行重命名或删除某个场景，场景命名应该遵循一定的规则，场景名称能反映场景动作，如图 4-8 所示。

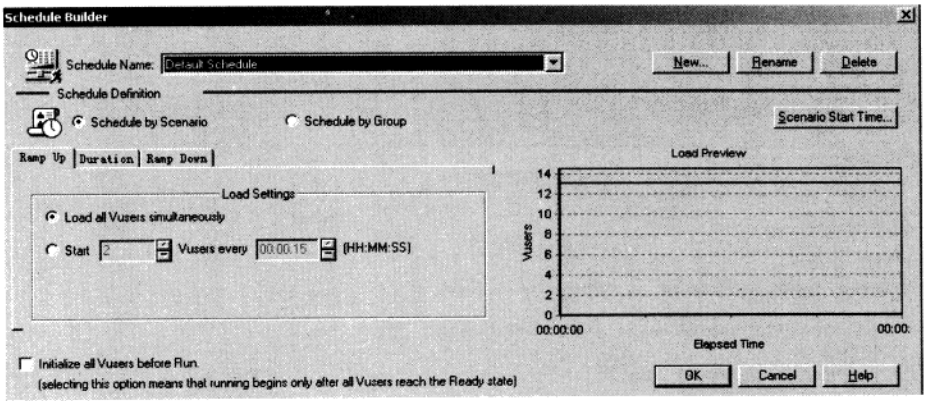


图 4-7 配置 Schedule

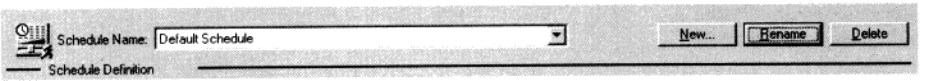


图 4-8 Schedule Name 设置

2. 按场景计划（Schedule by Scenario）

- Ramp Up 选项卡：设置场景虚拟用户加载方式，如图 4-9 所示。包含两种加载方式：
  方式一：场景执行时即将所有虚拟用户都加载完成。
  方式二：每隔一段时间加载一定的虚拟用户，通常采用这种方式来加载虚拟用户。

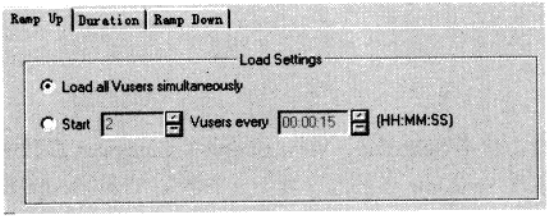


图 4-9 Ramp Up 选项卡

- Duration 选项卡：设置场景持续运行的情况，如图 4-10 所示。

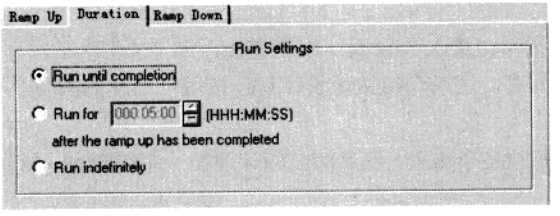


图 4-10 Duration 选项卡

持续运行方式一：按照设置运行，即每个虚拟用户按照指定的迭代次数运行，直到运行结束。

持续运行方式二：按指定时间运行，所有虚拟用户一直重复运行脚本，直到指定的时间结束运行，脚本迭代次数被忽略，不起作用。

持续运行方式三：一直运行，不停止，直到人为停止场景的运行才结束。

默认情况使用第一种方式，这是测试过程中最常使用的方式，第二和第三种方式常用于稳定性测试。

- Ramp Down 选项卡：设置场景执行完成后虚拟用户如何释放的策略，如图 4-11 所示。需要注意只有在 Duration 选项卡中设置为“按指定时间运行”时，才需要设置该项。

释放方式一：持续运行结束后，同时停止所有的虚拟用户。

释放方式二：每隔一段时间就停止一定量的虚拟用户，这项和 Ramp Up 中的第二项一样，区别在于前者是控制虚拟用户加载，后者是控制虚拟用户释放。

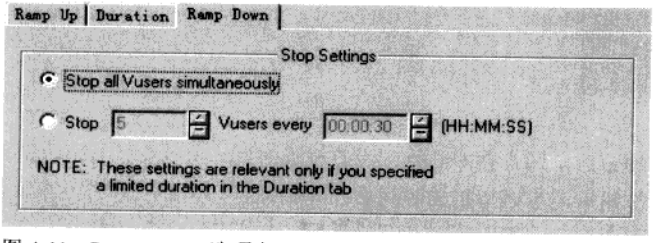


图 4-11 Ramp Down 选项卡

3. 按用户组计划（Schedule by Group）

按用户组计划设计场景比按场景计划设计场景的设置项中多出了 Start Time 选项卡，在按用户组计划设计场景中，以组为单位进行计划，每个组都要设置自己的 Start Time、Ramp Up、Duration 和 Ramp Down。

按用户组计划方式更加灵活，能够创建实际应用中脚本与脚本之间的约束关系。如一组用户执行后产生的数据记录为另一组用户的输入，这种情况就需要使用“用户组计划”方式来配置场景。

Ramp Up、Duration 和 Ramp Down 选项卡的内容与按场景计划设计场景中的内容一致，故只对 Start Time 选项卡进行分析，Start Time 选项卡内容如图 4-12 所示。

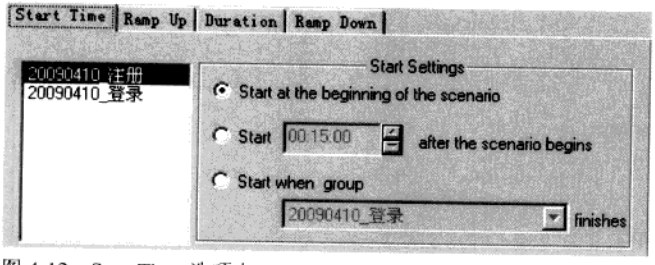


图 4-12 Start Time 选项卡

运行方式一：场景执行时立即开始运行该脚本。

运行方式二：场景执行一段时间后才开始运行该脚本。

运行方式三：在某个特定的用户组运行结束后才开始运行该脚本，通俗的讲就是在某个脚本运行结束后才开始运行。

4. 场景开始时间（Scenario Start Time）

如图 4-13 所示，场景启动时间设置有三种方式。

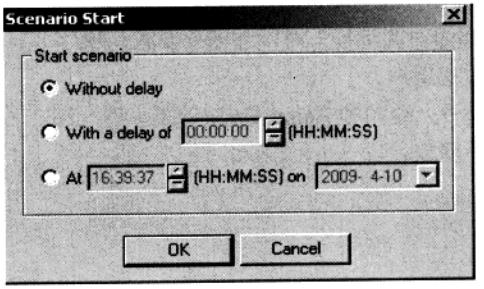


图 4-13 Scenario Start Time 设置

启动方式一：点击 Start Scenario 按钮后，场景立即开始，没有延误时间。

启动方式二：点击 Start Scenario 按钮后，推迟指定的时间后才开始运行。

启动方式三：点击 Start Scenario 按钮后，在指定的时间开始运行，如下午 16:30 运行。

5. 百分比模式

百分比模式是先设定好虚拟用户总数，然后按百分比的形式对所有的虚拟用户进行分配。该场景适合业务模型明确的性能测试。

百分比模式与“Manual Scenario（手动测试场景）”非常类似，但两者间还是存在一些不一致的地方。

如图 4-14 所示，在 Total Number of Vusers 文本框中设置虚拟用数总数。

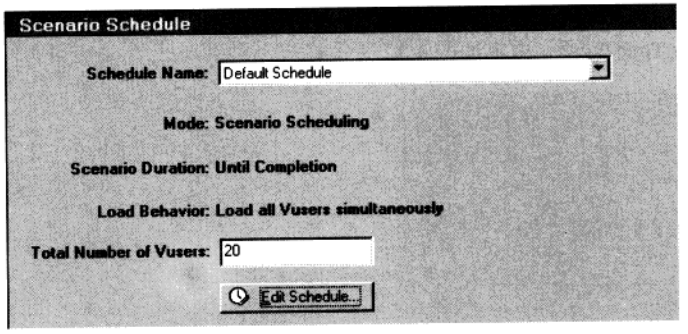


图 4-14 百分比模式场景计划信息

如图 4-15 所示，在“%”列中设置虚拟用户组所占总虚拟用户数的百分比。

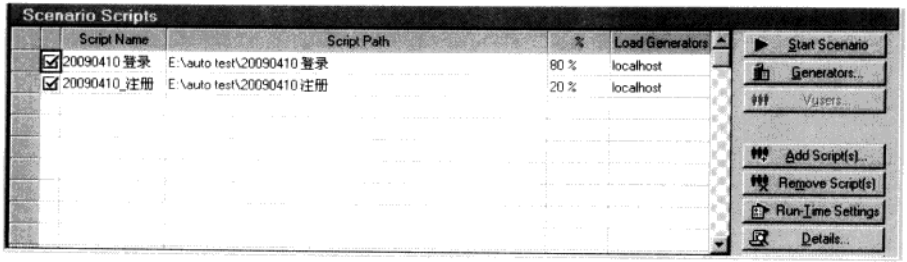


图 4-15 百分比模式中虚拟用户数设置

6. *初始化虚拟用户*

选中“Initialize all vusers before Run（初始化所有虚拟用户）”选项，表示必须等所有的虚拟用户都准备好之后，场景才能执行，如图 4-16 所示。

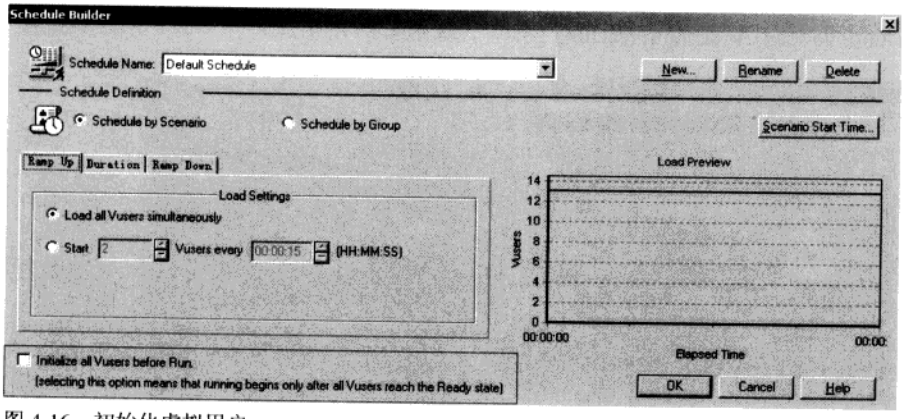


图 4-16 初始化虚拟用户

4.2.2 面向目标场景 Schedule 配置

在面向目标场景中，首先定义测试需要达到的目标，然后 LoadRunner 会自动根据这一目标创建场景。

在“新建场景”对话框中，选择“面向目标场景（Goal-Oriented Scenario）”，如图 4-17 所示。

在场景设置界面，点击 Edit Scenario Goal 按钮，进入编辑该目标场景对话框，包含五种目标类型，以 Hits per Second 目标类型为例，讲述其各项设置。

1. Scenario Settings 选项卡

在 Run Time 中设置一个时间值，表示当执行达到目标后，该场景还会持续运行一段时间（设置的时间值）才结束运行。

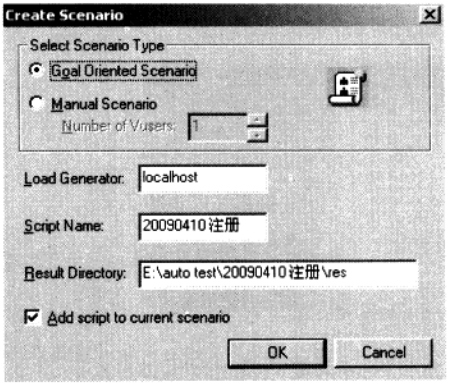


图 4-17　启动面向目标场景

If target cannot be reached 设置表示如果目标无法达到，Controller 将如何处理场景。有两种选择，可以选择“停止运行场景并保存结果（Stop Scenario and save results）”，或“继续运行场景直到达到目标（Continue scenario without reaching goal）”，如图 4-18 所示。

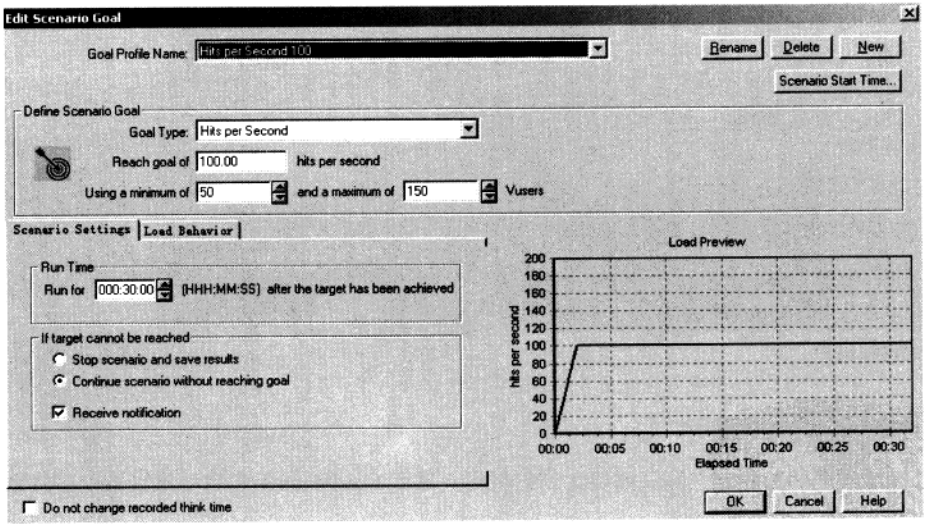


图 4-18　Scenario Settings 选项卡

2. Load Behavior 选项卡

设置加载行为，如图 4-19 所示。

加载行为一：让 Controller 自动加载用户。

加载行为二：设定一个时间，在该时间后达到目标。

加载行为三：每隔一段时间增加一定的目标量。

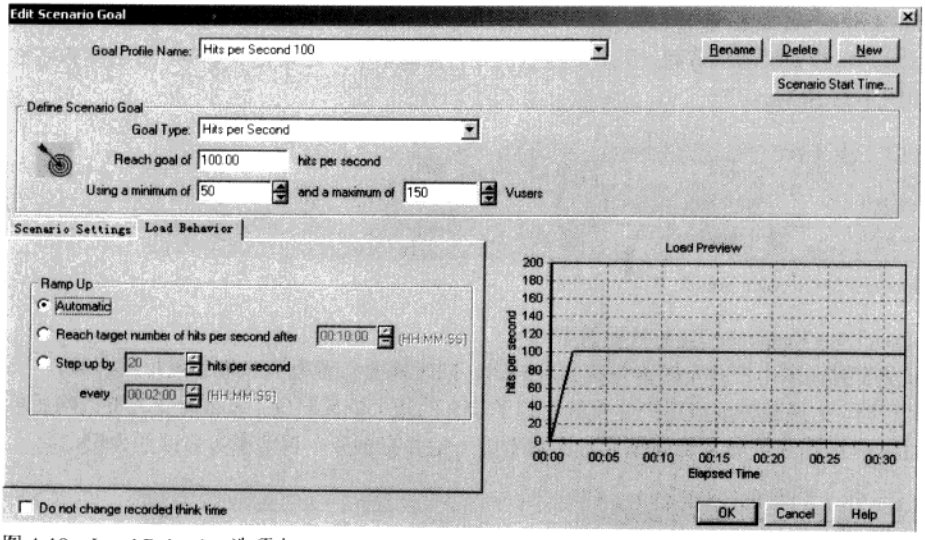


图 4-19 Load Behavior 选项卡

3. 目标类型（Goal Type）

（1）Virtual Users 目标类型

这种目标类型主要是用来测试服务器对并发用户的处理能力，这种目标类型与手动设置类型相类似，如图 4-20 所示。

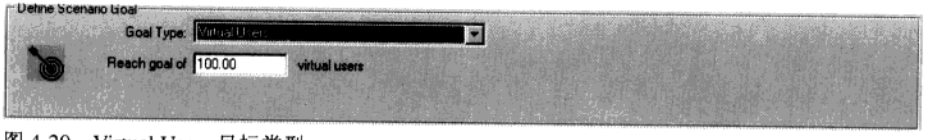


图 4-20 Virtual Users 目标类型

（2）Hits per Second 目标类型

设置的目标是点击数/秒。同时要设置最小虚拟用户数和最大虚拟用户数，当场景执行时，Controller 会使用最小的虚拟用户来达到定义的目标。如果使用最小的用户达不到定义的目标，那么 Controller 将会增加虚拟用户数，直到加载的虚拟用户数达到最大的虚拟用户数，如果使用最大的虚拟用户数还是达不到定义的目标，那么需要重新定义最大虚拟用户数，重新执行场景，如图 4-21 所示。

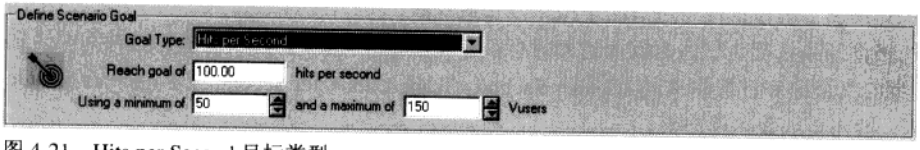


图 4-21 Hits per Second 目标类型

（3）Transactions per Second 目标类型

设置的目标为每秒处理的事务数，但需要注意的是，在脚本中一定要定义事务，否则事务名称栏为空白，如图 4-22 所示。

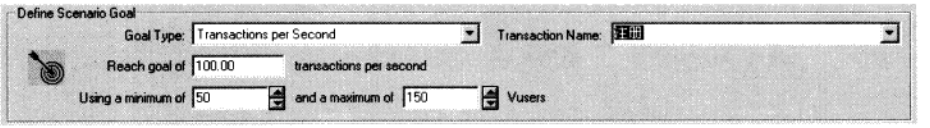


图 4-22 Transactions per Second 目标类型

（4）Transactions Response Time 目标类型

这类目标是设置在多用户并发时事务的响应时间。同时需要设置最大和最小虚拟用户数。

在场景执行时，如果部分虚拟用户并发就达到了定义的最大响应时间，那么需要调整策略后再重新执行，如图 4-23 所示。该目标类型要求在脚本中一定要有事务，否则事务名称栏为空白。

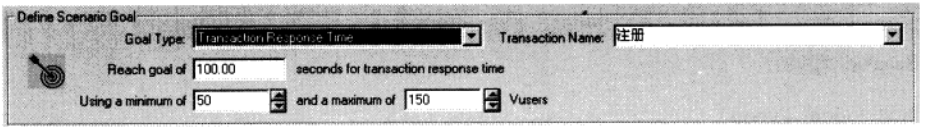


图 4-23 Transactions Response Time 目标类型

（5）Pages per Minute 目标类型

设置目标为每分钟处理的页面数，如图 4-24 所示。

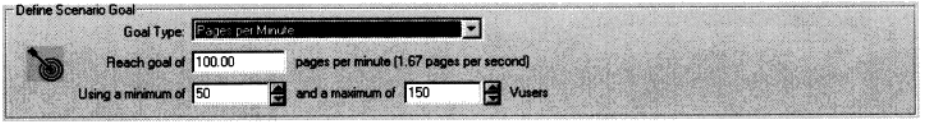


图 4-24 Pages per Minute 目标类型

如果定义的类型为 Hits per Second、Transactions per Second 或 Pages per Minute，控制器首先使用最小用户数除以定义的目标数，得到一值，通过这个值来确定每个用户应该达到的 Hits per Second、Transactions per Second 或 Pages per Minute 的值，接下来控制器将按 Load Behavior 选项卡中的设置加载虚拟用户。

1）如果选择控制器自动加载虚拟用户，控制器首先会加载 50 个虚拟用户。如果定义的最大虚拟用户数小于 50，那么会一次性将所有的虚拟用户加载完成。

2）如果选择在场景执行后一段时间内达到目标，那么 LoadRunner 会尝试在定义的时间内达到目标，并且根据时间限制和计算出来的 Hits per Second、Transactions per Second 或 Pages per Minute 的值来确定第一批需要加载的用户数。

3）如果选择一段时间内增加一定的目标量，那么 LoadRunner 会计算每个用户达到的值后，再

确定下一批需要加载的虚拟用户数。

每次加载一批用户后，LoadRunner 会判断是否达到目标值，如果没有达到预期目标值，LoadRunner 会重新计算每个虚拟用户应该达到的目标数，再重新调整下一批加载用户的数量。如果控制器加载到最多数量的虚拟用户还没有达到预定的目标，那么 LoadRunner 会重新计算每个用户的目标值，然后使用最大的用户数量去执行场景，尝试达到预期的目标。

在以下情况下，Hits per Second、Transactions per Second 和 Pages per Minute 类型的场景结果中会被置 Failed 状态：

1）控制器使用指定的最大用户数，并且执行两次都没有达到目标。

2）负载机不够。

3）所有的用户都运行失败。

4）控制器增加了几批虚拟用户后，Hits per Second、Transactions per Second、Pages per Minute 的值没有增加。

4.2.3 配置 View Script

在场景设计界面，用户脚本加载后，如果需要对加载的脚本进行修改，可以选中需要配置的脚本，点击右侧的 View Script 按钮，如图 4-25 所示。需要注意的是，如果对脚本修改后，一定要在 Controller 中重新加载该脚本，才能确保场景执行中的脚本是修改后的脚本。

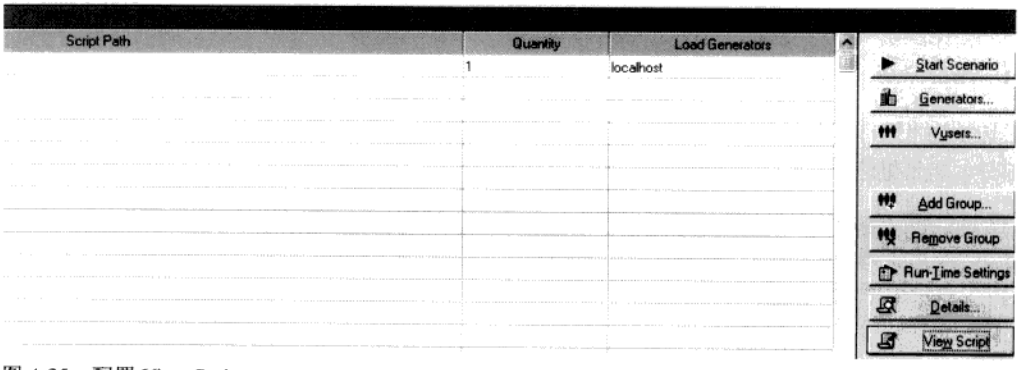


图 4-25 配置 View Script

4.2.4 配置 Load Generator

Load Generator 又叫负载发生器，当控制器发出执行命令时，Load Generator 负责和其他的负载机建立起联系并强制负载机执行。一般情况下在一台机器上安装好 LoadRunner，会自动安装好 Load Generator。一个 Controller 可以通过 Load Generator 来控制多台负载机。

点击右边的 Generators 按钮，会弹出 Load Generators 对话框，如图 4-26 所示。

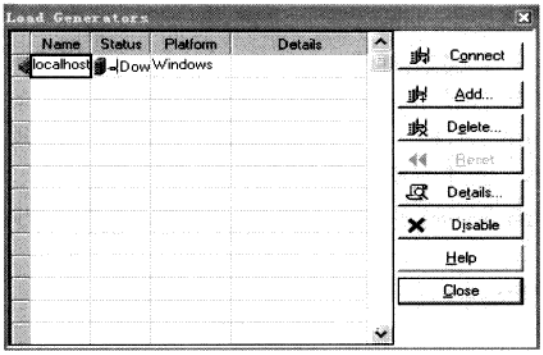


图 4-26 Load Generator 设置

在 Load Generators 对话框中，点击 Add 按钮，可以添加负载机，添加完成后，点击 Connect 按钮，测试负载机与控制机连接的情况，如果 Status 为 Ready，表示连接成功；如果为 Failed，表示连接失败，这时就要检查网络是否存在问题。

4.3 场景执行

场景设计完成之后，接下来要执行场景，在场景执行过程中要关注场景的运行情况，主要包括三个对象：场景、Vuser 组和 Vuser。

4.3.1 场景控制

切换到“运行”选项卡，在“运行”选项卡中主要包括两部分：场景组运行控制信息和数据图两部分。场景组运行控制信息如图 4-27 所示。

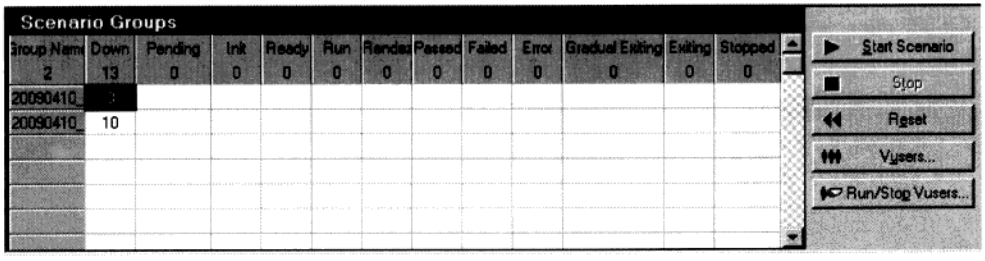


图 4-27 场景组信息

场景组的左边显示每个用户组的运行状态，右边为场景的控制操作，包括开始场景、场景停止、场景复位、虚拟用户控制和运行/停止 Vuser。

Start Scenario（开始场景）：点击该按钮，场景即开始运行。此时 Controller 开始初始化虚拟用户并将这些虚拟用户服务分配到负载发生器，开始运行脚本。

Stop（停止场景）：在场景未开始运行时，该按钮为灰，不可用，只有当场景已经开始运行后，该按钮才是可用状态。在运行期间点击该按钮场景将停止运行。对于如何控制场景停止运行有三种方式。

点击 Tools 菜单下的 Options 选项，弹出 Options 对话框，如图 4-28 所示。选中 Run-Time Settings 选项卡，有三种设置方式。

方式一：等当前迭代运行结束后，再停止运行场景（Wait for the current iteration to end before stopping）。

方式二：等当前的 Action 运行结束后，再停止运行场景（Wait for the current action to end before stopping）。

方式三：不等待，立即停止运行场景（Stop immediately）。

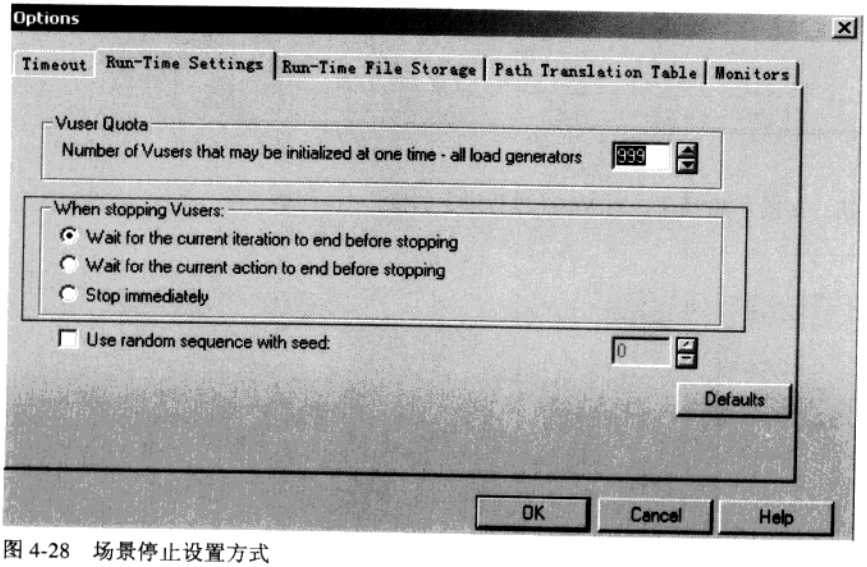


图 4-28 场景停止设置方式

Reset（重置/复位）：将方案中所有的 Vuser 组重置为方案运行前的“关闭”（Down）状态，准备下一次场景的执行。

Vusers...（虚拟用户组）：点击该按钮，能打开 Vuser 对话框，在 Vuser 对话框中可以查看 Vuser 组中每个 Vuser 的详细状态，如图 4-29 所示。显示出该组中每个 Vuser 的 ID、运行状态、脚本、负载发生器和所用时间。

在这里可以选择单个 Vuser 进行以下操作：

- 选择单个 Vuser 点击 Run 按钮，来运行该 Vuser。
- 选择单个 Vuser 点击 Stop 按钮，可以停止该 Vuser 的运行，点击 Tools 菜单→Options 选项→Run-Time Settings 选项卡，如图 4-28 所示，可以设置 Vuser 以哪种方式停止运行，

包括“等当前迭代运行结束后，再停止运行场景”或“等当前的 Action 运行结束后，再停止运行场景”两种方式。如果希望逐渐停止处于“运行”状态的 Vuser，则点击 Gradual Stop 按钮，该 Vuser 的状态将变为“正在逐步退出”并逐渐退出场景。

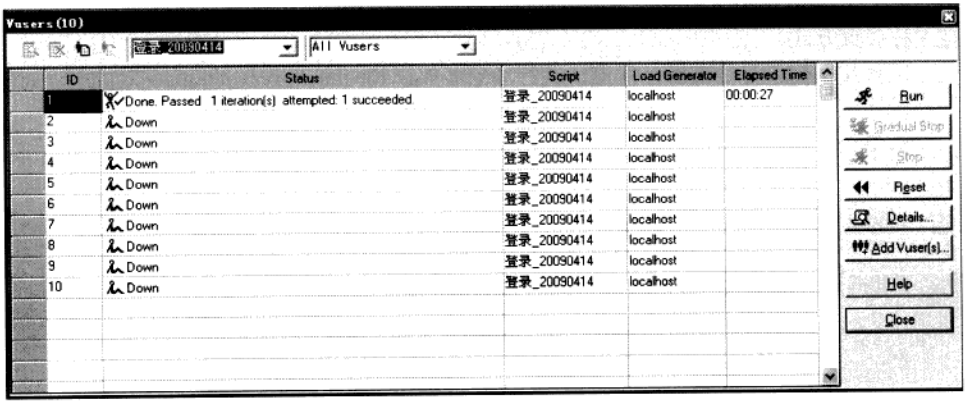


图 4-29　虚拟用户组

也可以右键点击，如图 4-30 所示，对 Vuser 进行以下的操作：

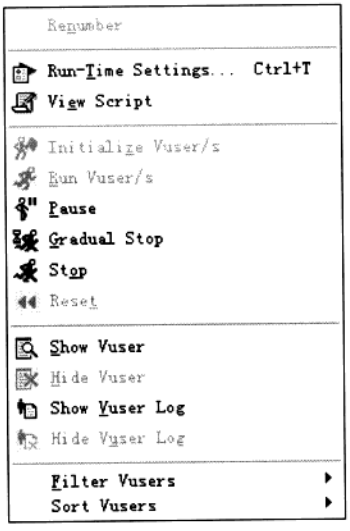


图 4-30　控制单个 Vuser

- 选择 Pause，可以暂停该 Vuser，但是暂停 Vuser 将影响其事务响应时间。
- 选择 Reset，可以重置该 Vuser，使其重新回到“关闭”的状态。
- 选择 Initialize Vuser/s，可以初始化该 Vuser。

- 选择 Renumber，可以对该 Vuser 编号重新定义。
- 选择 Filter Vusers，筛选显示出来的 Vuser，可以选择不同的筛选条件进行筛选，也可以在 Vuser 对话框的筛选器中选择需要使用的筛选条件。
- 选择 Sort Vusers，选择不同的排序方式对 Vuser 进行排序。
- 选择 Show Vusers，查看正在执行所分配脚本的 Vuser。此时会弹出运行查看器，并返回到 Vuser 的页面快照中，查看正在执行脚本的 Vuser。运行查看器的功能与浏览器的功能不同，查看器显示的图像是快照，而不是回放的所有特征。
- 选择 Show Vuser Log，会显示出该 Vuser 的脚本日志，如图 4-31 所示。

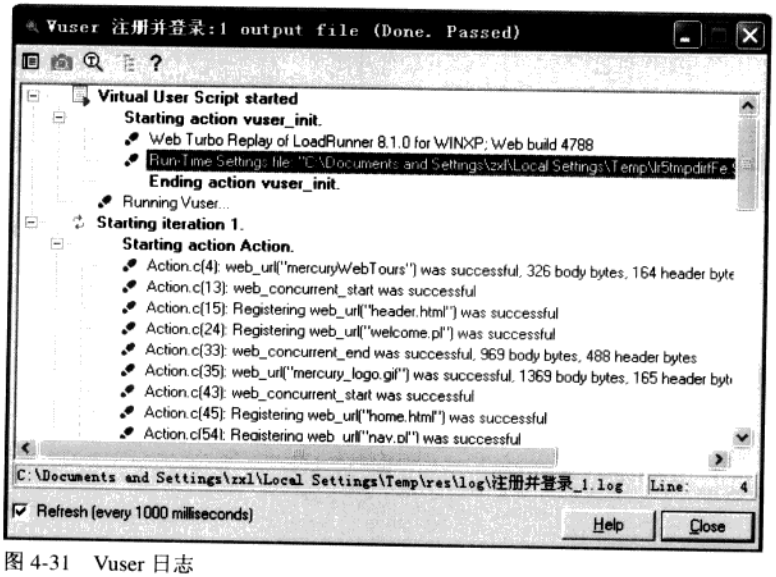


图 4-31 Vuser 日志

Run/Stop Vusers...（运行/停止 Vusers）：点击该按钮，打开 Run/Stop Vusers 对话框，在这里可以设置是继续执行还是停止某个用户组，如图 4-32 所示。

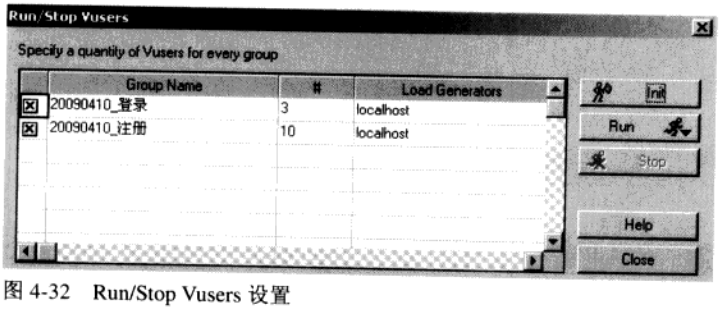


图 4-32 Run/Stop Vusers 设置

在运行期间，可以在 Run/Stop Vusers 对话框中手动控制新添加的 Vuser。该对话框因运行场景的模式不同而有所不同。

在手动模式下，可以控制添加到每个 Vuser 组中的 Vuser 数，以及运行这些添加 Vuser 的负载发生器，如图 4-33 所示。

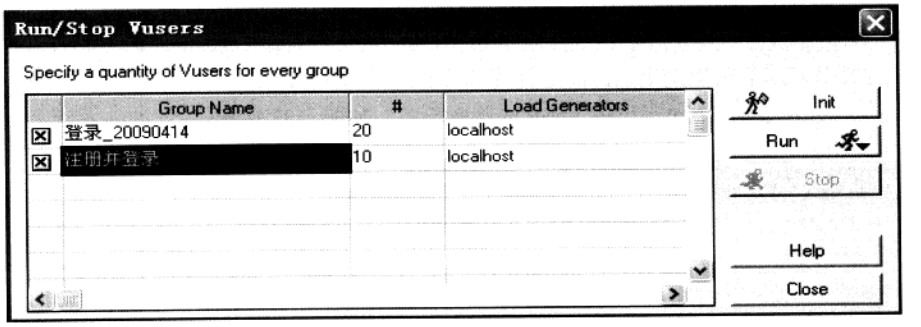


图 4-33　手动模式 Run/Stop Vusers 设置

也可以点击场景组中的 Vuser 按钮，在弹出的虚拟用户列表对话框中点击右侧控制区域的 Add Vuser(s)按钮，弹出“Add Vusers（添加虚拟用户）”对话框，如图 4-34 所示，可以在此对话框中添加虚拟用户信息。

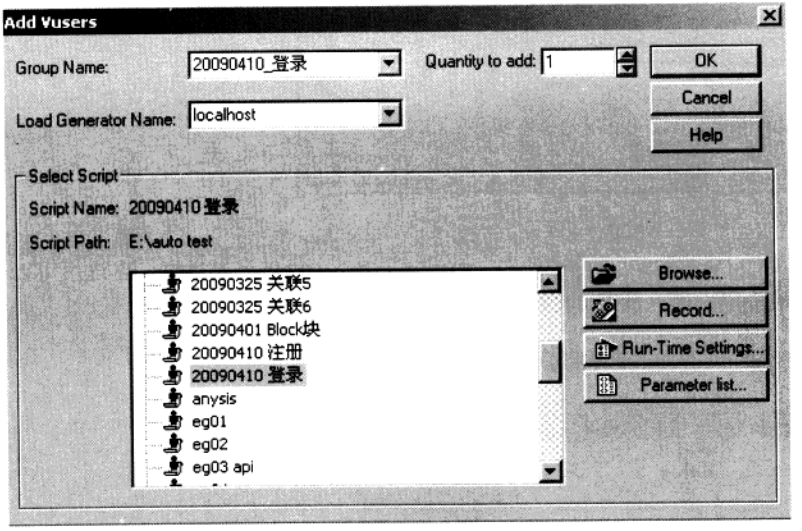


图 4-34　添加虚拟用户信息

如果在百分比模式下运行场景，能够根据定义的百分比，在 Vuser 脚本中分配新的 Vuser 数，以及运行这些添加的 Vuser 的负载发生器，如图 4-35 所示。

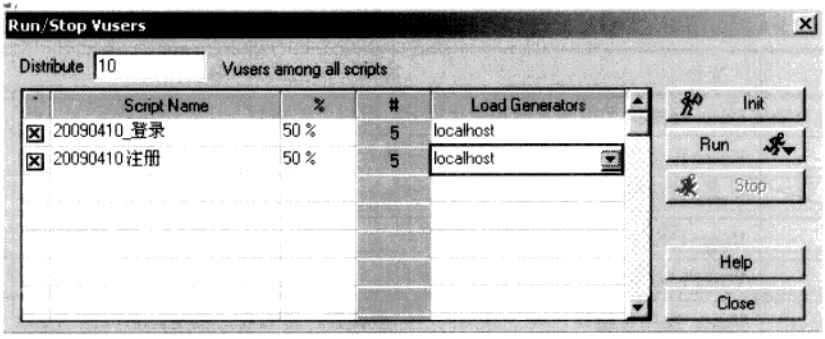


图 4-35 百分比模式 Run/Stop Vusers 设置

4.3.2 场景执行期间查看场景

在场景运行过程中，可以查看 Vuser 执行的情况，主要包括以下几个方面的内容：

- Vuser 运行状态。
- 事务详细信息。
- 查看“输出”窗口。

1. Vuser 运行状态

在场景执行期间，可以在“运行”视图中的“场景组”窗口中查看所有 Vuser 和 Vuser 组运行的状态，如图 4-36 所示。

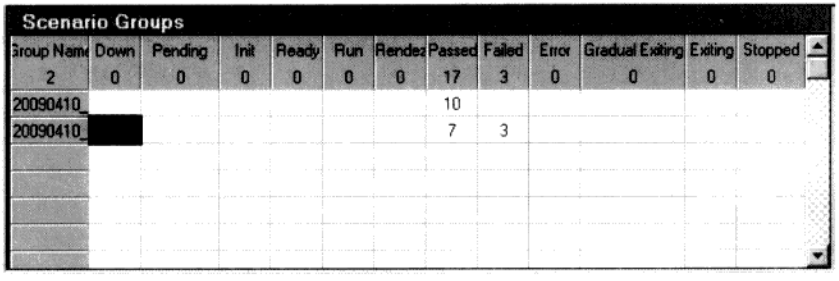


图 4-36 场景组中各 Vuser 组状态信息

场景组中这些状态的含义如表 4-1 所示。

表 4-1 虚拟用户组状态含义

状态	含义
关闭（Down）	Vuser 处于关闭状态
挂起（Pending）	Vuser 初始化已就绪，正在等待可用的负载发生器，或者正在向负载发生器传输文件
初始化（Init）	Vuser 正在进行初始化

续表

状态	含义
就绪（Ready）	Vuser 已经执行了脚本的初始化部分，可以开始运行
正在运行（Run）	Vuser 正在运行。正在负载发生器上执行 Vuser 脚本
集合点（Rendezvous）	Vuser 已经到达了集合点，正在等待释放
完成并通过（Passed）	Vuser 已结束运行。脚本执行通过
完成但失败（Failed）	Vuser 已结束运行。脚本执行失败
错误（Error）	Vuser 发生了错误。要了解错误的完整说明，请查看“输出”窗口或 Vuser 对话框中的“状态”字段
逐步退出（Gradula Exiting）	Vuser 正在运行退出前的最后一次迭代
退出（Exiting）	Vuser 运行结束，正在退出
停止（Stopped）	设置“停止”命令后，Vuser 立即停止

2. 事务详细信息

可以在“运行”视图右上角的框中查看正在运行场景的概况，如图 4-37 所示。

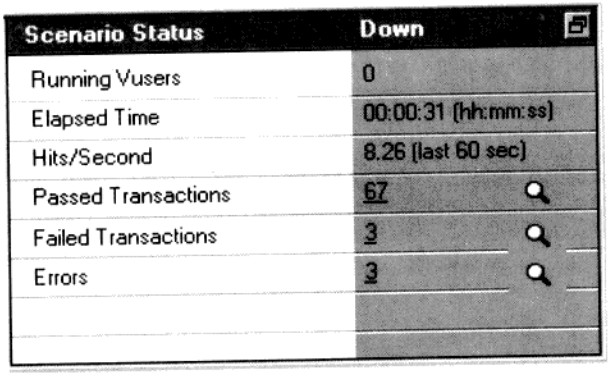


图 4-37 场景概要信息

点击右上角的按钮可以将“场景状态”窗口从“运行”视图中分离出来。这将放大“场景组”窗格。

“场景状态”窗口中各参数项的含义见表 4-2。

表 4-2 场景状态信息含义

状态概要	含义
场景状态（Scenario Status）	表示场景处于“正在运行”或“关闭”状态
正在运行的 Vuser（Running Vusers）	负载发生器计算机上正在执行的 Vuser 数
已用时间（Elapsed Time）	指自场景开始运行到现在所用的时间

续表

状态概要	含义
每秒点击次数（Hits/Second）	指每个 Vuser 运行期间，每秒对所测试服务器的点击次数（HTTP 请求数）
通过的事务数	场景运行到现在成功通过的事务数
失败的事务数	场景运行到现在失败的事务数
错误数	场景运行到现在发生错误的 Vuser 数

可以点击“场景状态”窗口中“通过的事务数”或“失败的事务数”右侧的“显示快照”按钮，弹出“事务”对话框，在该对话框可以看到事务的详细信息，如图 4-38 所示。

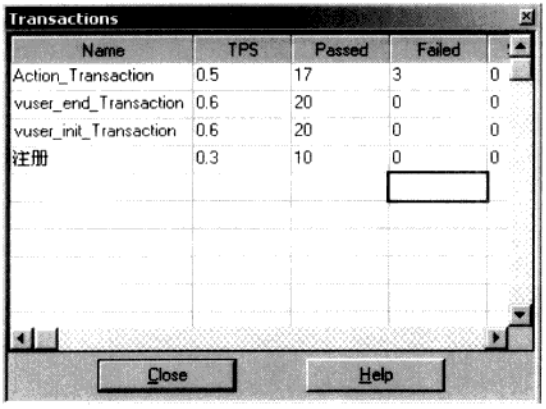


图 4-38 事务详细信息

“事务”对话框中各列值的含义如下：

名称（Name）：显示脚本中所有事务的名称。

TPS：表示每秒的事务数。

通过数（Passed）：表示运行已通过的事务数。

失败（Failed）：表示运行已失败的事务数。

停止（Stopped）：表示运行已停止的事务数。

3. 查看“输出”窗口

在场景运行时，Vuser 和负载发生器会向 Controller 发送错误、通知、警告、调试和批处理消息，这些信息可以在“输出”窗口中查看到。

在开始执行场景时，LoadRunner 会消除“输出”窗口中的消息。但是如果重置场景，消息还会保留在“输出”窗口中，除非设置 LoadRunner 在重置场景时删除“输出”窗口中的消息。

选择 View→Show Output，或者点击场景状态错误列表右侧的“显示快照”按钮。弹出“输出”对话框，如图 4-39 所示。该对话框默认显示的是错误信息。

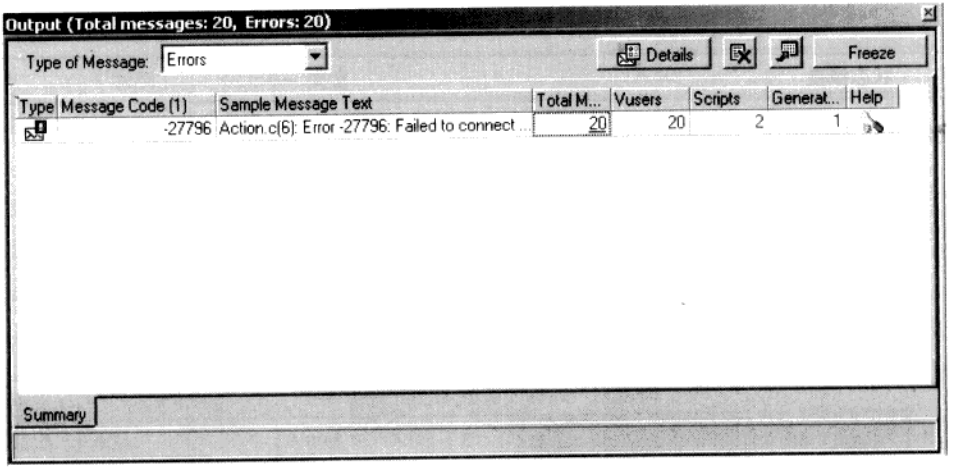


图 4-39 场景消息输出窗口

在“Type of Message（消息类型）”下拉列表框中，可以选择要显示的消息类型。

当运行中存在错误信息时，为了查看到更加详细的信息，可以选择一条信息并点击“Details（详细信息）”按钮，“输出”对话框中则会打开“Detailed Message Text（详细消息文本）”框，在这里会显示出完整的消息文本示例，如图 4-40 所示。

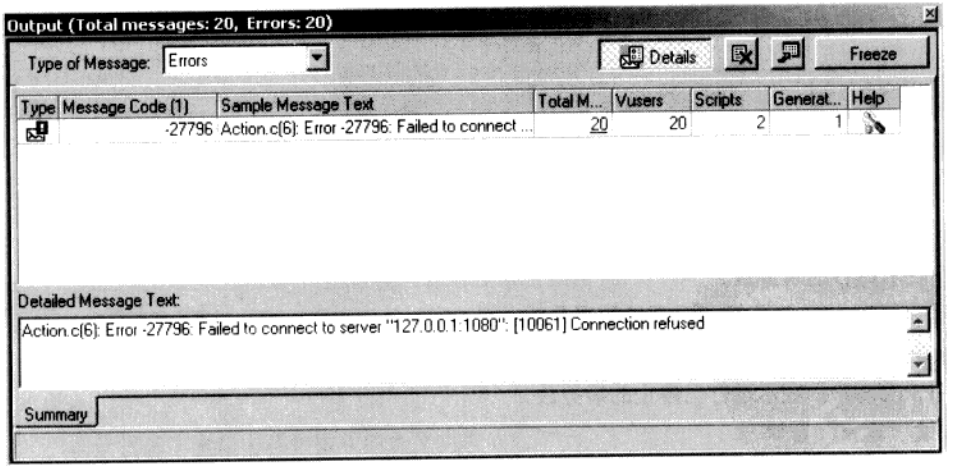


图 4-40 查看详细错误信息

一条错误消息可能与虚拟用户、脚本和负载发生器有关。如果需要查看更加详细的信息，可以点击相应列的蓝色链接，如图 4-41 所示。

图 4-41 显示，错误发生在 20090410 注册脚本中，在第一次迭代，第 4 行代码中出错，负载机为 localhost。

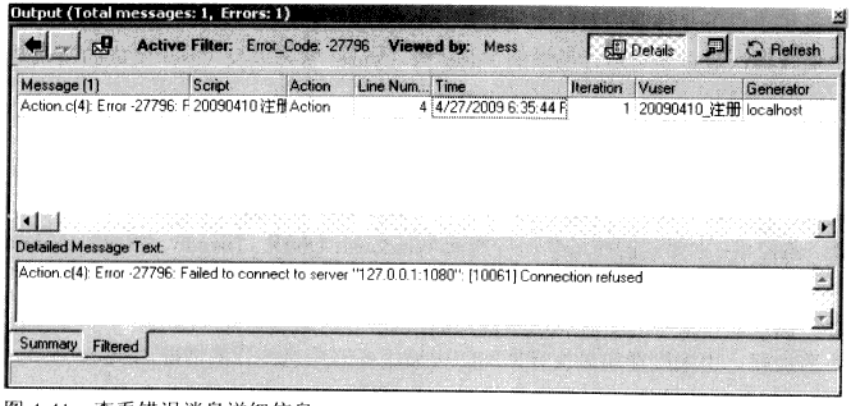


图 4-41 查看错误消息详细信息

4.4 场景监视

在前面的场景信息中介绍了如何观察虚拟用户运行的状态，通过日志文件可以追溯虚拟用户运行失败是由脚本中哪条语句引起的。但是仅仅依靠这些信息还是无法更好地分析性能瓶颈，因为性能测试最重要的是要找到服务器性能瓶颈出现的原因及调优方法，因此，需要通过监视器来获得更多的数据，帮助分析服务器性能瓶颈。所以要了解如何添加监视器和分析监视曲线图。

4.4.1 关于联机监控

通过联机监控器，LoadRunner 可以查看场景执行期间生成的数据。使用 LoadRunner 联机图，可以指定场景执行期间 Controller 要监控的计算机，并可以查看监控器收集的数据。

在监控场景之前，必须安装和配置 LoadRunner 监控组件。LoadRunner 监控的整个过程，如图 4-42 所示。

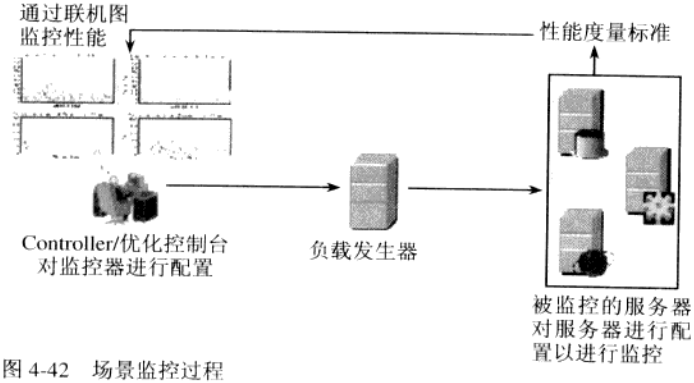


图 4-42 场景监控过程

LoadRunner 主要提供了以下几种监视器：

- “运行时”监视器：显示参与运行场景的 Vuser 数、Vuser 状态以及 Vuser 生成的错误数和类型。
- “事务”监视器：显示场景运行时各事务速率和响应时间。
- “Web 资源”监控器：监视场景运行期间 Web 服务器上的信息，主要包括 Web 连接数、吞吐量、HTTP 响应数、服务器重试次数和下载到服务器的页面数等信息。
- “系统资源”监控器：主要是监控场景运行期间 Windows、UNIX、Tuxedo、SNMP、Antara FlameThrower 和 SiteScope 资源使用情况。
- “网络延迟”监控器：显示关于系统网络延迟的信息。
- “防火墙”监控器：用来度量场景执行期间防火墙服务器信息统计的情况。
- “Web 服务器资源”监控器：用来度量场景运行期间 Apache、Microsoft IIS、iPlanet（SNMP）和 iPlanet/Netscape Web 服务器的统计信息。
- “Web 应用程序服务器资源”监控器：度量场景运行期间应用程序服务器 Ariba、ATG Dynamo、BroadVision、ColdFusion、Fujitsu INTERSTAGE、iPlanet（NAS）、Microsoft ASP、Oracle9iAS HTTP、SilverStream、WebLogic（SNMP）、WebLogic（JMX）和 WebSphere 统计信息的情况。
- “数据库服务器资源”监控器：用于度量场景运行期间数据库 DB2、Oracle、SQL 服务器和 Sybase 统计信息的情况。
- “流媒体”监控器：用来度量场景运行期间 RealPlayer 和 Media Player 客户端以及 Windows Media 服务器和 RealPlayer 音频/视频服务器的统计信息..
- “ERP/CRM 服务器资源”监控器：用来度量场景执行期间 SAP R/3 系统、SAP Portal、Siebel Server Manager、Siebel Web 服务器和 PeopleSoft（Tuxedo）服务器的统计信息。
- “Java 性能”监控器：用于度量 J2EE 对象及 J2EE 和 EJB 服务器对象的统计信息。
- “应用程序组件”监控器：用来度量场景执行期间 Microsoft COM+和 Microsoft .NET CLR 服务器的统计信息。
- “应用程序部署解决方案”监控器：用来度量场景执行期间 Citrix MetaFrame XP 和 1.8 服务器的统计信息。
- “中间件性能”监控器：度量场景执行期间 Tuxedo 和 IBM WebSphere MQ 服务器的统计信息。
- “基础结构资源”监控器：用于度量场景执行期间网络客户端数据点的统计信息。

4.4.2 监控器与度量

在场景执行过程中，可以监视各服务器的运行情况，在监视服务器之前还要做一些工作来确保监视连接成功。

1）修改被监视主机访问模式：进入“管理工具”→“本地安全策略”→“安全选项”→“网

络访问：本地账户的共享和安全模式”。将访问方式更改为“经典－本地用户以自己的身份验证”，如图 4-43 所示。默认情况下选择的访问方式为“仅来宾－本地用户以来宾身份验证”。如果设置为“来宾”模式访问，即以 Guest 来访问，而 Guest 账户没有监控的权限，这样在监控的过程中会一直提示无法连接服务器。还有一个需要注意的地方是，一定要设置密码，否则在 Monitor Configuration 中添加 Measurements 仍然会提示拒绝登录。

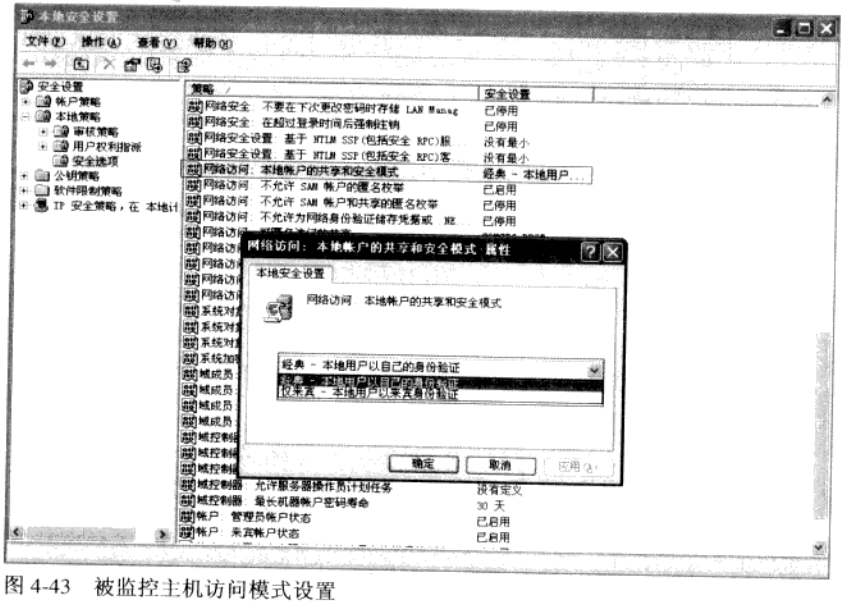


图 4-43 被监控主机访问模式设置

2）保证被监视系统开启以下三个服务 Remote Procedure Call（RPC）和 Remote Registry Service、Remote Registry。其中 Remote Procedure Call（RPC）Locator 的登录选项中要输入当前主机账户和密码，然后重启该服务，其他服务设置不变。注意：有时只要开启两个服务 Remote Procedure Call（RPC）和 Remote Registry Service 就可以，但是 Windows XP 必须开启 Remote Registry 这个服务。

3）确认安装 Controller 的机器可以连接到被监视的机器。如果监控失败，并且 LoadRunner 找不到指定的服务器，请确认指定的服务器是否可用。在 Controller 或优化控制台计算机命令行中键入 ping <server_name>（或 ip），执行 ping 操作。

4）验证可以正常连接之后，如果仍有其他问题，可以参考表 4-3 的解决方法。

5）确认并打开共享文件。首先，在被监视的机器上右击“我的电脑”→“选择管理”→“共享文件夹”→“共享”，在这里要有 C$这样一个共享文件夹。该文件可能存在，也可能不存在，没有的话，需要手动增加。然后，在安装 LoadRunner 的机器上使用“运行”，在命令行中输入“\\被监视机器 IP\C$ ”，然后输入管理员账号和密码，如果能看到被监视机器的 C 盘了，就说明 LoadRunner 所在计算机具有被监视机器的管理员权限，可以使用 LoadRunner 去连接了。

表 4-3 监视器连接问题解决方案

问题	解决方案
无法监控其他域中的 Windows 计算机，或者“访问被拒绝”	要获得对远程计算机的管理权限，请在命令提示符下执行以下命令：%net use \\<计算机名>/用户:[<域>\<远程计算机名>]，提示输入密码时，输入远程计算机的密码
无法监控 Windows NT/Windows 2000 计算机（发出一条错误消息“未找到计算机名”或“无法连接到主机”）	要监控的 Windows NT/Windows 2000 计算机仅允许具有管理员权限的用户进行监控。要允许非管理员用户进行监控，必须授予用户对特定文件和注册表项的读取权限（Microsoft 技术说明编号 Q158438）。需要执行下列步骤： a）使用浏览器或文件管理器，授予用户对下列项的读取权限： %windir%\system32\PERFCxxx.DAT %windir%\system32\PERFHxxx.DAT 其中 xxx 是系统的基本语言 ID，例如，英语的 ID 为 009。这些文件可能已丢失或损坏。如果对此有怀疑，请从安装 CD 中提取这些文件。 b）使用 regedt32，授予用户对下列项的读取权限：HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Perflib c）使用 regedt32，至少授予用户对下列项的读取权限：HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurePipeServers\winreg
无法从 Windows NT 计算机监控某些 Windows 2000 计数器	在 Windows 2000 计算机上运行 Controller 或优化控制台
某些 Windows 默认计数器生成错误无法从被监控的计算机上获得 SQL Server 6.5 版的性能计数器	删除有问题的计数器，并使用“添加度量”对话框添加相应计数器。这是 SQL Server 6.5 版的一个错误。解决方法为：在被监控的计算机上使用 regedt32，授予用户对以下注册表项的读取权限：HKEY_LOCAL_MACHINE\Software\Microsoft\MSSQLServer\MSSQLServer（Microsoft 技术说明编号 Q170394）
选定的度量未显示在图中	确保已注册显示文件和 online.exe。要在不执行完全安装的情况下注册监控器的 dll，请运行 LoadRunner\bin 中的 set_mon.bat 批处理文件
监控 Windows 计算机时，图中不显示任何度量	检查内置的 Windows 性能监控器。如果该监控器不能正常工作，则可能是通信设置有问题
监控 UNIX 计算机时，图中不显示任何度量	确保 rstatd 正在 UNIX 计算机上运行（请参阅“系统资源监控”）
无法监控下列 Web 服务器之一：Microsoft IIS、Microsoft ASP 或 ColdFusion	要获得对远程计算机的管理权限，请在命令提示符下执行以下命令：%net use \\<计算机名>/用户：[<域>\<远程计算机名>]提示输入密码时，输入远程计算机的密码
无法监控 WebLogic（JMX）服务器	打开<LoadRunner 根文件夹>\dat\monitors\WebLogicMon.ini 文件，并搜索：[WebLogicMonitor] JVM=javaw.exe，将 javaw.exe 更改为 java.exe，将打开一个包含跟踪信息的窗口

接下来要在 Controller 控制器中添加要监控的计算机资源。

选择菜单 Monitors→Add Measurements 或在视图中点击鼠标右键选择 Add Measurements，如图 4-44 所示。

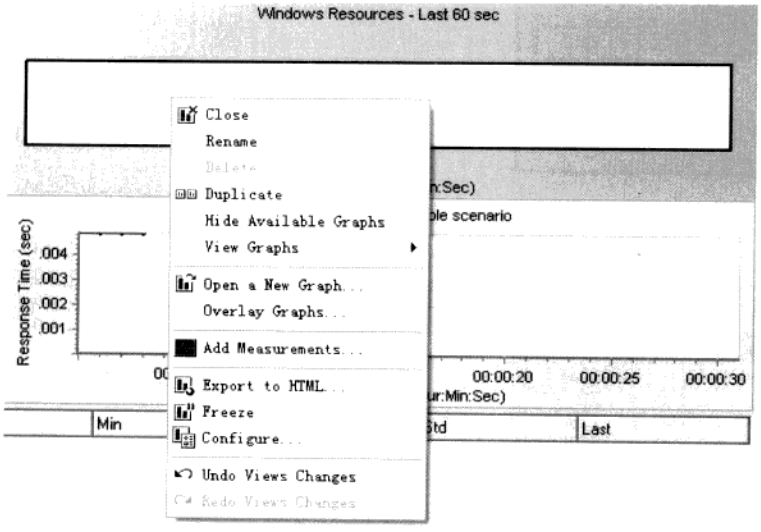


图 4-44 添加度量

弹出系统资源对话框，如图 4-45 所示。

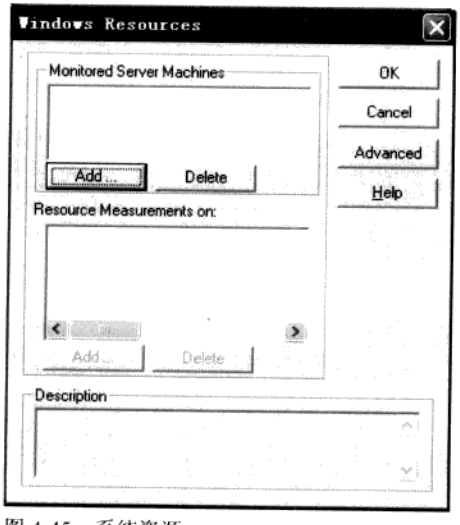


图 4-45 系统资源

在该对话框点击 Add 按钮，可以添加要监控的计算机，如图 4-46 所示。在该对话框中输入要监控的计算机的机器名或 IP 地址和选择操作系统平台。

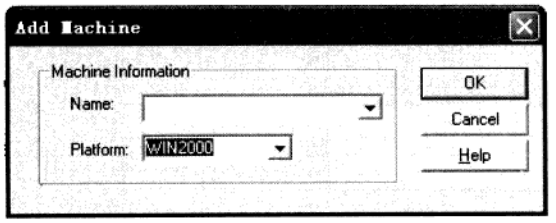


图 4-46 添加机器信息

添加好要监控的计算机后，可以在“资源度量位于:<计算机>”中选择要监视的一些指标进行添加，如图 4-47 所示。

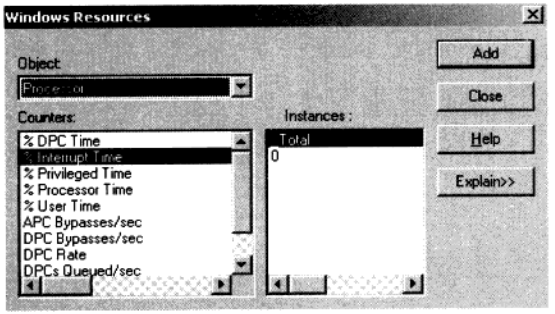


图 4-47 添加监控服务器的资源度量指标

4.4.3 联机监视器

在场景运行期间，可以在监控视图中查看数据变化的情况。默认情况下显示运行 Vuser、事务响应时间、每秒点击次数和 Windows 资源 4 个视图，如图 4-48 所示。

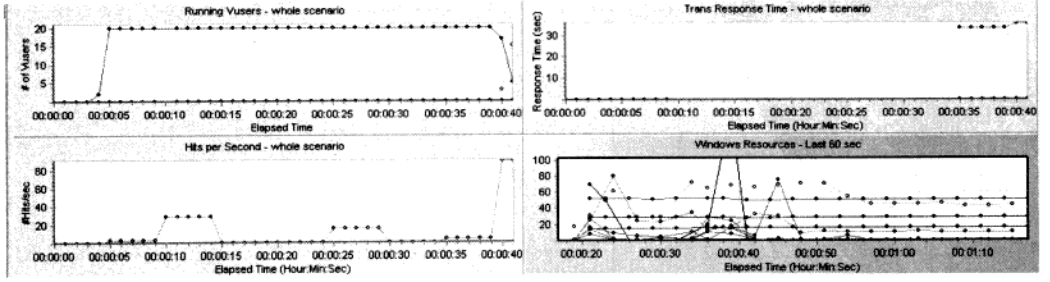


图 4-48 默认视图

可以双击某个视图，将其最大化，再次双击又还原为平铺视图。

同时显示视图的个数也可以设置，点击鼠标右键选择 View Graphs 项，可以设置同时显示 1 个、2 个、4 个或 8 个视图，也可以选择自定义同时显示视图的个数，但自定义显示视图个数最多只能设置 16 个视图同时显示，如图 4-49 所示。

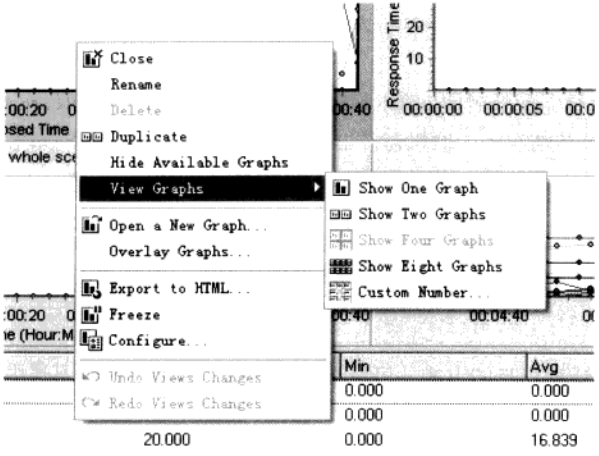


图 4-49 显示视图数设置

关于监视器选择设置。在菜单 Tool→Options 中的 Monitors 选项卡中设置。使用该选项卡可以启用事务监控器，配置事务数据的行为，并设置联机监控器的数据采样速率、错误处理、调试和频率，如图 4-50 所示。

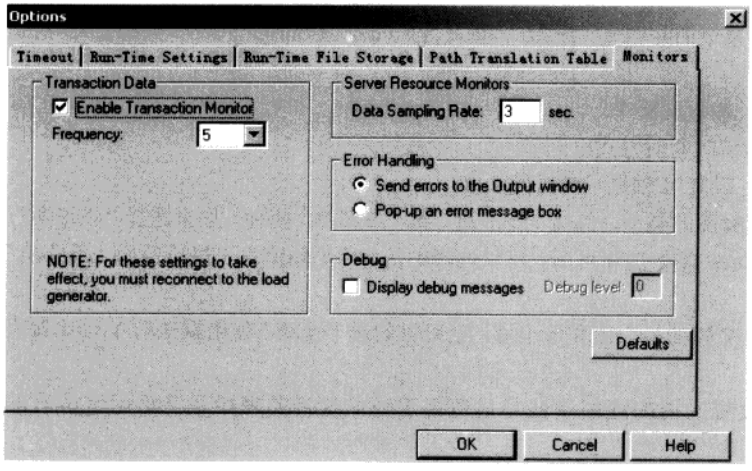


图 4-50 监视器选项设置

事务数据：用来配置“事务”、“数据点”、“Web 资源”联机图的数据行为。要注意的一点是，更改设置后，一定要重新连接负载机后才能生效。

- 启用事务监视器：使联机的 Vuser 事务监控器在场景开始时监控事务。
- 频率：用来设置联机监控器为生成“事务”、“数据点”、“Web 资源”联机图采集数据的频率，以秒为单位。默认值为 5 秒。对于小型场景，建议设置频率为 1 秒；对于大型场景，建议设置为 3～5 秒。频率越高，网络流量越低。在这个指定的时间段内，系统将计算数据的平均值，并仅向 Controller 发送一个值。
- 数据采样速率：采样速率是连续采样之间的时间间隔，以秒为单位。用来设置 LoadRunner 在场景中采集监控数据的速率。默认情况下，联机监控器采集数据的间隔为 3 秒。如果增大采样速率，则数据的监控频率就会降低。此设置将应用于所有的视图。

设置的数据采样速率将应用于随后激活的所有监控器，但并不应用于已激活的服务器监控器。要对已激活的服务器监控器应用新的数据采样速率，请保存该场景，然后重新打开它。

错误处理：用来控制 LoadRunner 发出错误消息的方式。可以选择发送至“输出”窗口或弹出错误消息框中的一种。

- 将错误发送至“输出”窗口：将执行场景过程中所有的错误发送至“输出”窗口。
- 弹出错误消息框：把错误发送至消息框。

调试：用来调试场景。

- 显示调试信息：把与调试有关的信息发送至输出日志。可以指定一个调试级别，调试级别可选择范围为 1～9。其中调试级别仅与网络监控有关。

在场景执行过程中为了更好地监控视图可以对视图的属性进行设置。选择“监控器”→“联机图”→“配置”，或者，右击某个图并选择“配置”，将打开“Graph Configuration（图配置）”对话框，如图 4-51 所示。

- Refresh rate（sec）：输入刷新率。
- Time：设置 X 轴显示时间的方式。
- Graph Time（sec）：设置 X 轴显示的时间长度。
- Display Type：设置视图的类型，包括“线形图”和“条形图”两种。如果选择“条形图”，那么下面的 Bar Values Type 选项可用，可供选择的值有“平均值”、“最后值”、“最小值”或“最大值”。
- Y-Axis Scale：设置 Y 轴的最大值和最小值，也可以勾选“自动”使用默认的 Y 轴比例查看图。
- 设置适用范围：设置当前的视图属性是只适用于当前选择的视图还是适用于所有的视图。

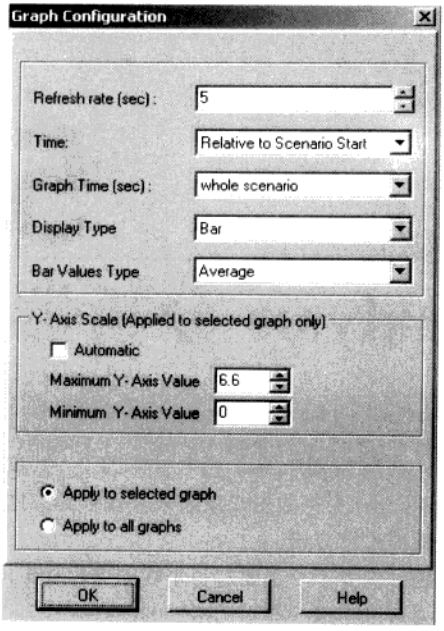


图 4-51 设置监视器属性

5 Analysis 分析器

分析器顾名思义就是对测试结果数据进行分析的组件，它是 LoadRunner 三大组件之一，其重要性不言而喻。在 Controller 组件执行场景的过程中，LoadRunner 会将数据收集起来并保存到数据库中。当场景执行完成后，可以进入 Analysis 组件对这些数据进行分析。

分析器中保存着大量用来分析性能测试结果的数据视图，但并不一定要对每个视图进行分析，可以根据实际情况选择相关的数据视图进行分析，分析结果可以生成一些不同格式的测试报告。

5.1 Analysis 简介

5.1.1 Analysis 基础知识

要分析系统瓶颈，就必须借助 LoadRunner 分析器中的数据来帮助分析。在场景执行过程中，LoadRunner 会收集执行过程中的数据，并将数据存储到结果文件中，其扩展名为.lrr。在 Analysis 分析器，打开保存的结果文件，Analysis 会对收集到的信息进行处理，并生成图和报告。

Analysis 会话至少包含一组方案结果（lrr 文件）。Analysis 会将活动图的显示信息和布局设置存储在扩展名为.lrr 的文件中。

关于数据分析，不仅仅局限于只在 Analysis 分析器中对数据进行分析，可以采用多种方式进行分析：

- Vuser 日志文件：Vuser 日志文件包括每个 Vuser 运行方案的完整跟踪。这些文件位于方案结果目录中。
- Controller 输出窗口：在 Controller 输出窗口会显示出整个方案运行过程中的错误或警告信息，当然其中最关注的信息还是输出的错误信息，通过查看这些错误信息有利于帮助性能调试工作。
- Analysis 图：通过使用一些分析技术对数据图表进行合并、关联等操作，更好地帮助分析

事务、Vuser 等其他的一些信息。

- “图数据”视图和“原始数据”视图以电子表格形式显示用于生成图的实际数据。当然也可以将这些数据复制到外部电子表格应用程序中进行处理。
- 报告形式：LoadRunner 自动带了多种格式的报告供分析，包括 HTML、Word 和水晶报表三种形式的报告。

那么 Analysis 中的数据是怎么得到的呢？其实在场景运行的时候，默认情况下，所有的 Vuser 信息都保存在该 Vuser 的负载机上。只有当场景运行结束之后，这些数据才会自动进行整理或合并，这时负载机上所有的 Vuser 的信息和数据都将被传输到结果目录中。默认情况下，在 Controller 控制器中，Results→Auto Collate Results 是被选中的，如图 5-1 所示。也就是说默认情况下，当场景运行结束后，这些数据会被自动整理或合并。当然如果不希望自动整理或合并这些数据，可以不选中这一项。但不选中这一项，并不代表不会对这些数据进行整理或合并，在 Analysis 分析器对这些数据进行分析之前，Analysis 会对这些数据进行整理。或者可以在 Controller 控制器中选择 Results →Collate Results 进行手动整理。

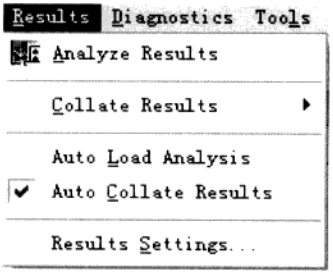


图 5-1 自动整理或合并结果设置

5.1.2 设置选项

在进行数据视图分析之前，有必要将分析图中涉及到的一些设置项设置完成。那么为什么在分析图之前要进行设置呢？原因很简单，当场景运行结束后，分析器收集到的所有数据都是原始的数据，未经过任何条件筛选的数据，这样就有很多“杂质”并不方便对数据进行分析。下面介绍一些常用的设置选项的设置。

1. Result Collection 设置

在 LoadRunner 处理这些数据时，可以查看这些数据的摘要。具体怎样查看摘要数据，需要在 Result Collection 选项中进行设置。选择 Tools→Options→Result Collection，如图 5-2 所示。

Data source:

- Generate summary data only：表示仅查看摘要数据。如果选择该选项，那么 Analysis 不会处理数据以用于筛选和分组等高级用途。注意，当选择该项时，Data Aggregation 项是不可以设置的。

- Generate complete data only：表示仅查看经过处理的完整数据，但是不显示摘要数据。
- Display summary while generating complete data：表示在处理完整数据时，查看摘要数据。

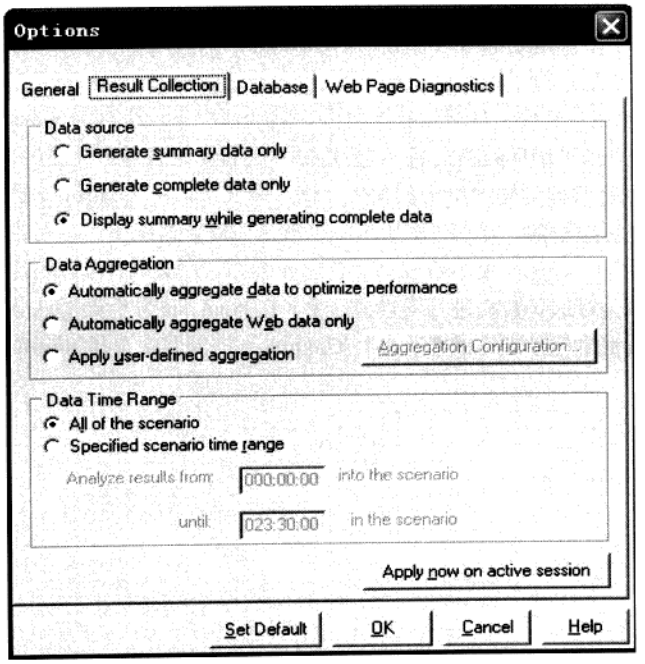


图 5-2 配置数据选项设置

Data Aggregation：

- Automatically aggregate data to optimize performance：表示使用内置数据聚合公式聚合数据，以优化性能。
- Automatically aggregate Web data only：表示使用内置数据聚合公式仅仅聚合与 Web 有关的数据。
- Apply user-defined aggregation：表示用户自定义来设置聚合数据。

点击 Aggregation Configuration... 按钮，会弹出聚合配置的对话框，如图 5-3 所示。这里有以下几个设置项：

Aggregate Data（available only for complete data）：用于设置聚合数据的类型，这里只适用于完整数据。可以选择需要聚合数据的图的类型，这样可以减小数据库的容量。可供选择数据图的类型有 Transactions、Web、Monitors、DataPoints 和 Script Errors 几种。

Select the graph properties to aggregate：指定要聚合的图属性，可选择的属性有 VuserID、Group Name、Script Name 和 Do not aggregate failed Vusers。

Web data aggregation only：表示仅聚合 Web 的数据，在这里可以设置聚合的粒度。

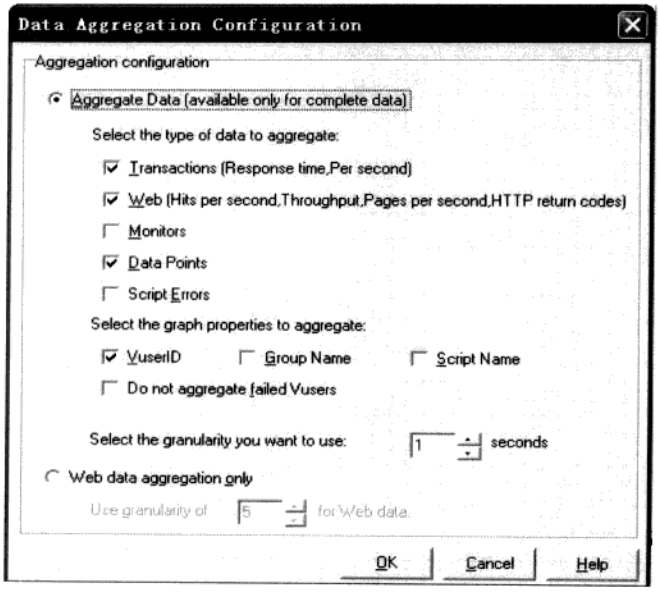


图 5-3 数据聚合配置

2. Set Granularity 设置

通过更改 X 轴的粒度（比例），可以使视图更方便于阅读和分析。最大粒度是图的时间范围的一半。为了确保可读性和清晰性，Analysis 在大于等于 500 秒的范围内自动调整图的最小粒度。

选择 View→Set Granularity 或直接点击 按钮，会弹出时间粒度设置对话框，如图 5-4 所示。

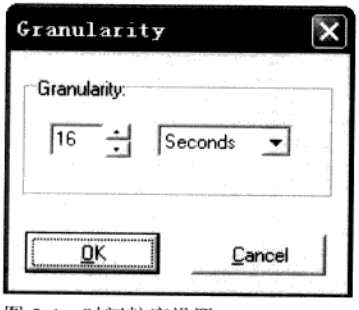


图 5-4 时间粒度设置

实例：使用不同的粒度来显示每秒点击次数图，Y 轴表示在粒度间隔内的每秒点击次数。如图 5-5 所示粒度为 5，Y 轴表示每隔 5 秒统计一次点击次数。图 5-6 所示粒度为 10，Y 轴表示每隔 10 秒统计一次点击次数。

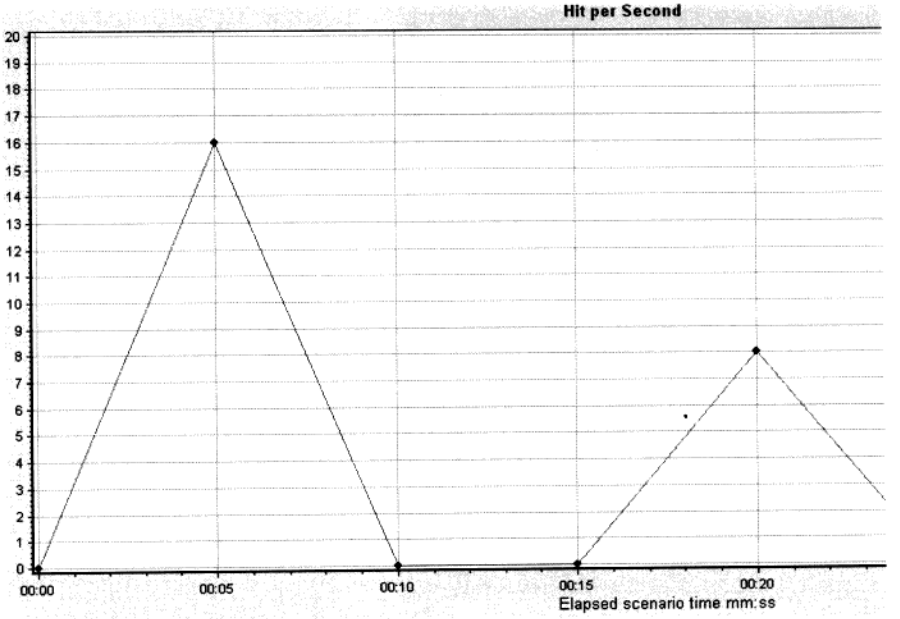


图 5-5　粒度为 5 时每秒点击次数图

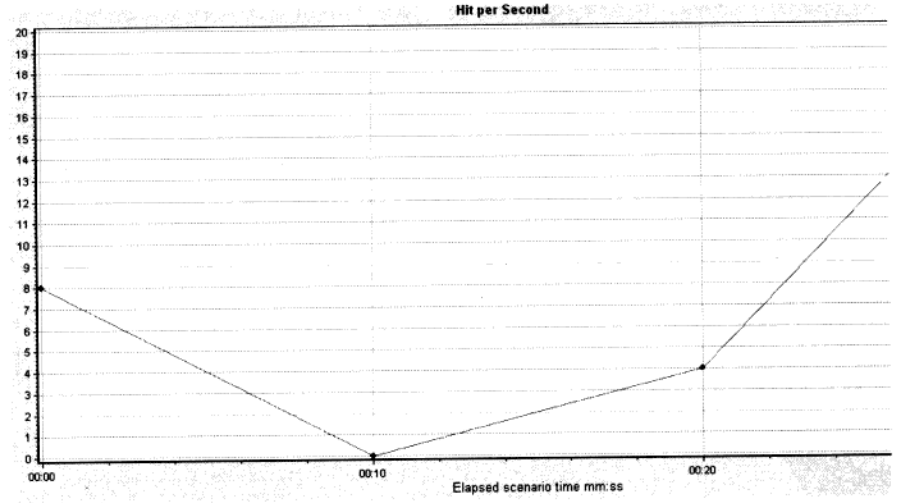


图 5-6　粒度为 10 时每秒点击次数图

3. Configure Measurements 设置

在分析图时，有时会发现分析图的 Y 轴幅值过小，这时就可以通过设置 Configure

Measurements，对 Y 轴进行适当的放大或缩小操作。

选择菜单 View→Configure Measurements 选项，弹出 Configure Measurements 设置对话框，如图 5-7 所示。

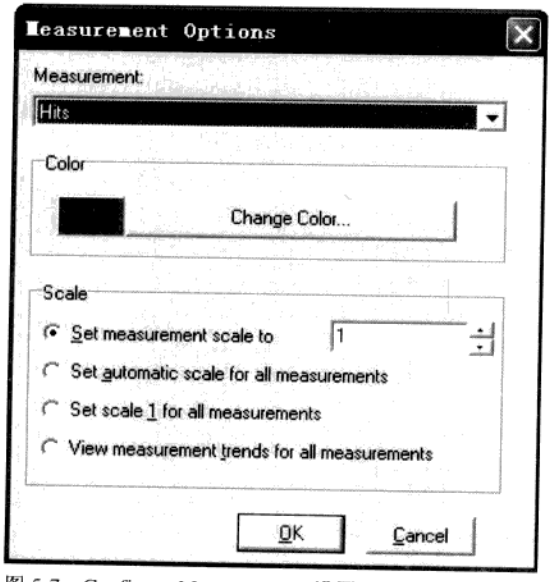


图 5-7 Configure Measurements 设置

在这里 Y 轴的比例设置有四种选择：

- Set measurement scale to：手动设置一个比例值。
- Set automatic scale for all measurements：使用优化的自动比例来显示图中每个度量。
- Set scale 1 for all measurements：将图中所有度量比例都设置为 1。
- View measurement trends for all measurements：查看所有度量的趋势。

4. 设置筛选条件

在结果分析的过程中，怎么提取最关键的数据，对分析结果是一个很重要的问题，在分析过程中可以对分析图中的数据进行筛选以提取最关键的数据。在 Analysis 分析器中，关于筛选有三种：如果只要设置单个图的筛选条件时，使用 Graph Settings 对话框，点击 按钮；如果要设置方案中所有图的筛选条件时，使用 Global Filter 对话框，点击 按钮；如果是摘要报告，那么使用 Analysis Summary Filter 对话框，点击 按钮。但这三种筛选设置都大同小异，本质是一样的，没什么大的区别，下面主要讲述 Graph Settings 对话框设置，如图 5-8 所示。

筛选条件：在这里要为每个筛选条件选择条件和值。

条件：可以选择“=”（等号）或“<>”（不等号）。

值：从 Values 栏列表中选择一个值。筛选条件的值有三种类型，包括离散、连续和基于时间。

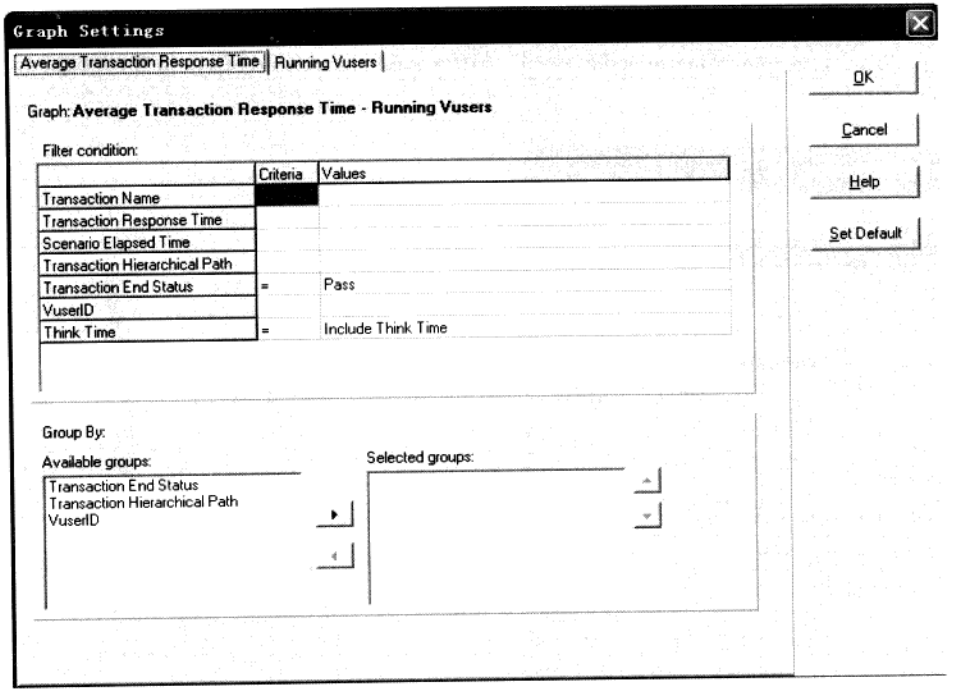


图 5-8 筛选条件设置

离散值是一个明确的整数值，如 Transaction Name 下拉列表中会显示出所有的事务名。

注：在这里可以使用 Transaction Hierarchical Path 事务父树路径条件来筛选子事务。选择“事务名”以筛选父级的子事务，选择“无”以筛选父事务，或者选择“未知”以筛选其父级未知（通常由会话期间的嵌套错误引起）的子事务。

- 连续值是一个变量维度，可以在最小值和最大值范围限制内取任何值，如 Transaction Response Time。
- 基于时间的值是基于方案开始时间的值。

分组方式：使用这些设置来对图显示按组排序。有可用组和选定组两个复选框。

需要注意 Think Time 维度的设置，在录制测试脚本过程中，如果执行脚本时没有忽略 Think Time 时间，那么 Analysis 分析器在统计分析结果时会把 Think Time 包含进去。这样当 Think Time 存在于用户事务的开始与结束之间时，相关事务统计情况会受到影响。因此，很多时候需要过滤用户的思考时间。默认情况下 Think Time 的值设置为 Include Think Time，将下拉复选框中该选项去掉即可，分析结果中就会自动滤掉思考时间。

5.1.3 Analysis 图

Analysis 分析器中提供了丰富的分析图，常见的有 8 类：Vusers 图、错误图、事务图、Web 资

源图、网页细分图、系统资源图、Web 服务器资源图和数据库服务器资源图。选择 Graph→Add Graph，如图 5-9 所示。

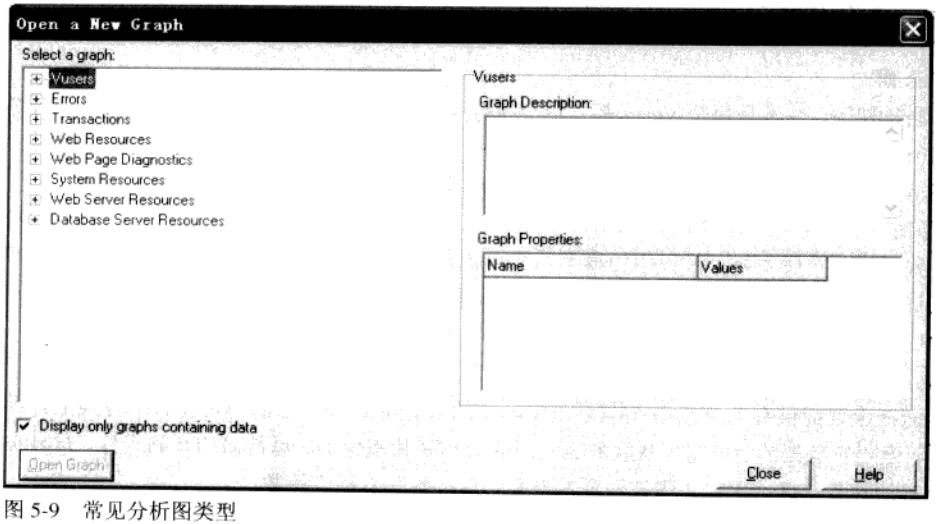


图 5-9 常见分析图类型

但是如果将图 5-9 中 Display only graphs containing data 复选框的勾选去掉，会发现多出了很多其他类型的分析图，如图 5-10 所示。

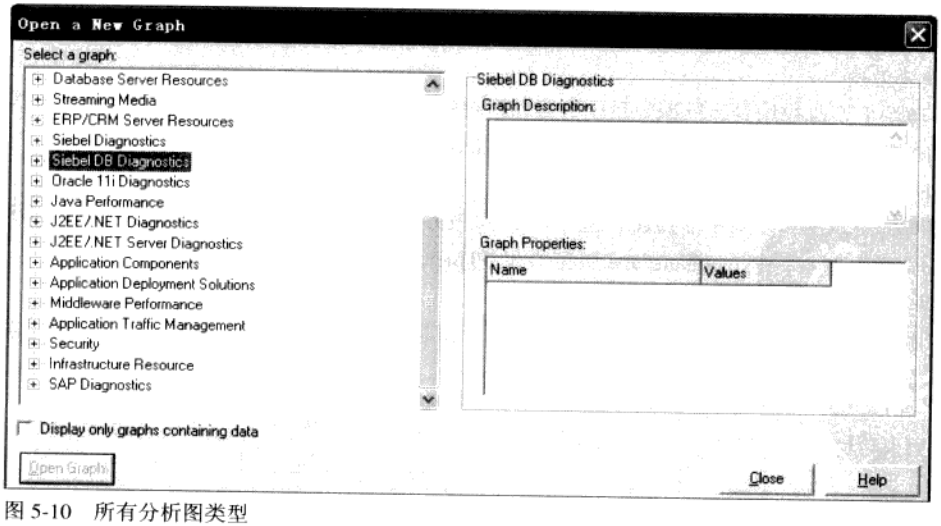


图 5-10 所有分析图类型

在实际分析的过程中，只有常见的几种分析图会使用到，其他的都很少使用到。

1. Vusers 图

在方案执行过程中，Vuser 在执行事务时生成数据。使用 Vusers 图可以确定方案执行期间 Vuser 的整体行为。它显示 Vuser 状态和完成脚本的 Vuser 的数量。主要包括正在运行的 Vuser 图和 Vuser 摘要图两种。

2. Errors 图

在方案执行期间，并不是每个 Vuser 都一定会执行成功，可能有执行失败、停止或因错误而终止的情况。Errors 图主要是统计方案执行时的错误信息。主要包括：Error Statistics（by Description）、Error per Second（by Description）的、Error Statistics 和 Error per Second 四种图。

3. 事务图

事务图描述了整个脚本执行过程中的事务性能和状态。主要包括：平均事务响应时间图、每秒事务数图、每秒事务总数、事务摘要图、事务性能摘要图、事务响应时间（负载下）图、事务响应时间（百分比）图和事务响应时间（分布）图。

4. Web 资源图

Web 资源图主要提供有关 Web 服务器性能的一些信息。使用 Web 资源图可以分析方案运行期间每秒点击次数、服务器的吞吐量、从服务器返回的 HTTP 状态代码、每秒 HTTP 响应数、每秒页面下载数、每秒服务器重试次数、服务器重试摘要、连接数和每秒连接数。

5. 网页细分图

网页细分图主要是提供一些信息来评估页面内容是否影响事务响应时间。如果出现影响事务响应时间的情况，可以通过细分图进一步分析是什么原因影响网页事务响应时间的。包括网页细分、页面组件细分、页面组件细分（随时间变化）、页面下载时间细分、页面下载时间细分（随时间变化）和已下载组件图几种。

6. 系统资源图

系统资源图主要监控场景运行期间系统资源使用率的情况。可以监控 Windows 资源、UNIX 资源、SNMP 资源、Antara FlameThrower 资源和 SiteScope 资源。

7. Web 服务器资源图

Web 服务器资源图主要用来捕捉场景运行时 Web 服务器的信息。主要用来分析 Microsoft IIS 服务器、Apache 服务器、iPlanet/Netscape 服务器和 iPlanet（SNMP）服务器。

8. 数据库服务器资源图

数据库服务器资源图主要显示数据库服务器的统计信息。目前支持 DB2、Oracle、SQL Server 和 Sybase 数据库。

5.2 摘要报告

摘要报告提供了场景执行的一般信息。该报告始终存在于树视图或者作为 Analysis 窗口中的选项卡。可以通过选择 View→Export Summary to Excel 将摘要报告导出到 Excel 中。主要包括概要、

统计、事务统计和 HTTP 响应统计四大部分。

5.2.1 概要部分

Analysis 概要总结部分的信息如图 5-11 所示。

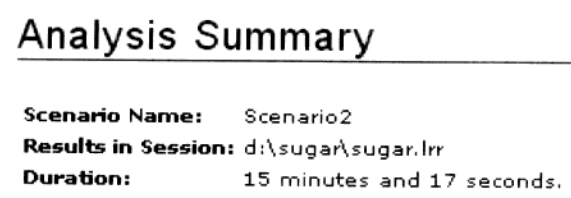


图 5-11 摘要图概要部分

包括三部分内容：

1. Scenario Name（场景名）

显示场景名的内容，如果该场景保存过，那么将会显示场景的保存路径。

2. Results in Session（场景结果）

显示场景结果文件存储的路径和结果文件名。

3. Duration（运行时间）

显示本场景运行的总时间，如果脚本中包含有 Think Time，那么会显示 Think Time 的时间。

5.2.2 统计部分

统计部分显示的信息如图 5-12 所示。

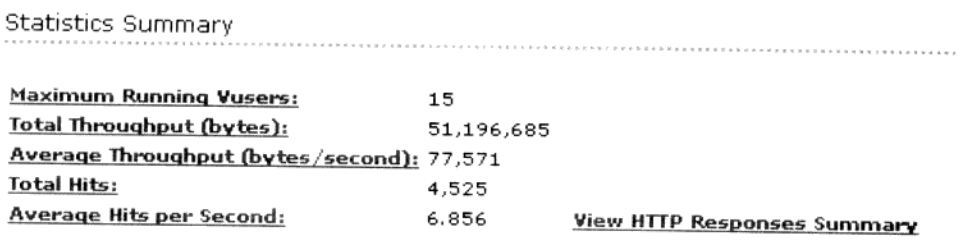


图 5-12 摘要图统计部分

包含 6 个链接内容：

1. Maximum Running Vusers（最大运行 Vuser 数）

显示的信息是场景运行时设置的 Vuser 用户数，但是发现这个数值往往比设置的 Vuser 数要小，这是因为 LoadRunner 有加载时间和延迟时间。这里在 Controller 中设置了 20 个虚拟用户，但在图中只显示为 15 个。

2. Total Throughput（bytes）（总吞吐量）

表示场景在运行时产生的全部网络流量，单位为字节。

3. Average Throughput（bytes/second）（平均吞吐量）

表示平均每秒的吞吐量，即吞吐率。

4. Total Hits（总点击数）

表示在场景运行期间，所有的 HTTP 请求总数。

5. Average Hits per Second（平均每秒点击数）

表示平均每秒的点击数。

6. View HTTP Responses Summary（查看 HTTP 响应摘要）

查看 HTTP 响应摘要。

5.2.3 事务统计部分

事务统计信息如图 5-13 所示。

Transaction Summary

Transactions: Total Passed: 286 Total Failed: 0 Total Stopped: 0　　Average Response Time

Transaction Name	Minimum	Average	Maximum	Std. Deviation	90 Percent	Pass	Fail	Stop
商业机会_提交	2.522	4.206	12.501	3.409	12.501	7	0	0
商业机会_进入界面	0.762	0.842	0.991	0.066	0.991	7	0	0
客户反馈_进入界面	1.513	1.731	2.488	0.245	2.063	22	0	0
客户档案_进入界面	0.762	0.866	1.368	0.153	1.029	25	0	0
潜在客户_提交	0.769	0.835	0.957	0.068	0.942	12	0	0
潜在客户_进入界面	0.776	0.979	1.376	0.185	1.24	11	0	0
营销活动_提交	0.753	1.247	3.85	0.438	1.408	101	0	0
营销活动_进入界面	0.784	1.243	6.274	0.544	1.401	101	0	0

图 5-13　摘要图事务统计部分

事务统计部分第一行统计场景运行时所有事务通过、失败和停止的数量。接下来是一个表格，在该表格中显示了所有事务执行时的一些较详细的信息，表格中每列的意思如下：

1. Transaction Name（事务名）

该列表示事务名。

2. Minimum

事务运行的最短时间。

3. Average

事务运行的平均时间。

4. Maximum

事务运行的最长时间。

5. Std.Deviation

标准方差，Std 是单词 standard 的缩写，方差描述一组数据偏离其平均值的情况。方差公式

如下：

$$S^2 = \frac{1}{n}\left[(x_1 - \overline{x})^2 + (x_2 - \overline{x})^2 + \cdots + (x_n - \overline{x})^2\right]$$

从数学角度来看，方差值越大，说明这组数据就越离散，波动性也就越强。方差越小，说明这组数据就越聚合，波动性也就越小。

6. 90 Percent

在 Controller 运行场景时，并不会显示这个值，因为它是对整个一系列数据统计的结果。表示一个事务在执行过程中的 90%所花费的时间，比如，一个事务执行了 100 次，90%表示其中 90 次事务运行时间的平均值。

7. Pass

通过的事务个数。

8. Fail

失败的事务个数。

9. Stop

停止的事务个数。在场景执行时，若用户手工停止场景的执行，事务没有自己的状态，那么就是停止状态。

5.2.4 HTTP 响应统计

HTTP 响应摘要如图 5-14 所示。

HTTP Responses Summary

HTTP Responses	Total	Per second
HTTP_200	4,196	6.358
HTTP_301	3	0.005
HTTP_302	321	0.486
HTTP_404	5	0.008

图 5-14 HTTP 响应摘要

此视图只有 Web Vuser 才有，它反映了 Web Server 的处理情况。

表格有三列，其各列含义如下：

1. HTTP Responses

这里表示 HTTP 响应的状态码。

2. Total

该 HTTP 响应状态码总的点击数。

3. Per second

该 HTTP 响应状态码每秒的点击数。

详细 HTTP 响应状态码的含义，见表 5-1。

表 5-1 HTTP 响应状态码

HTTP 状态码	含义
100	客户必须继续发出请求
101	客户要求服务器根据请求转换 HTTP 协议版本
200	交易成功
201	提示知道新文件的 URL
202	接受和处理、但处理未完成
203	返回信息不确定或不完整
204	请求收到，但返回信息为空
205	服务器完成了请求，用户代理必须复位当前已经浏览过的文件
206	服务器已经完成了部分用户的 GET 请求
300	请求的资源可在多处得到
301	删除请求数据
302	在其他地址发现了请求数据
303	建议客户访问其他 URL 或访问方式
304	客户端已经执行了 GET，但文件未变化
305	请求的资源必须从服务器指定的地址得到
306	前一版本 HTTP 中使用的代码，现行版本中不再使用
307	申明请求的资源临时性删除
400	错误请求，如语法错误
401	请求授权失败
402	保留有效 ChargeTo 头响应
403	请求不允许
404	没有发现文件、查询或 URL
405	用户在 Request-Line 字段定义的方法不允许
406	根据用户发送的 Accept，请求资源不可访问
407	类似 401，用户必须首先在代理服务器上得到授权
408	客户端没有在用户指定的时间内完成请求
409	对当前资源状态，请求不能完成
410	服务器上不再有此资源且无进一步的参考地址
411	服务器拒绝用户定义的 Content-Length 属性请求
412	一个或多个请求头字段在当前请求中错误
413	请求的资源大于服务器允许的大小
414	请求的资源 URL 长于服务器允许的长度
415	请求资源不支持请求项目格式

续表

HTTP 状态码	含义
416	请求中包含 Range 请求头字段，在当前请求资源范围内没有 Range 指示值，请求也不包含 If-Range 请求头字段
417	服务器不满足请求 Expect 头字段指定的期望值，如果是代理服务器，可能是下一级服务器不能满足请求
500	服务器产生内部错误
501	服务器不支持请求的函数
502	服务器暂时不可用，有时是为了防止发生系统过载
503	服务器过载或暂停维修
504	端口过载，服务器使用另一个端口或服务来响应用户，等待时间设定值较长
505	服务器不支持或拒绝请求头字段中指定的 HTTP 版本

5.3 Analysis 常见图分析

5.3.1 Vuser 图

在方案执行过程中，Vuser 在执行事务时生成数据。使用 Vusers 图可以确定方案执行期间 Vuser 的整体行为。它显示 Vuser 状态和完成脚本的 Vuser 的数量。将这些图与事务图结合使用可以确定 Vuser 的数量对事务响应时间产生的影响。

这里包含两种图：正在运行的 Vuser 图和 Vuser 摘要图。

正在运行的 Vuser 图显示在测试期间的每一秒内，执行 Vuser 脚本的 Vuser 的数量及它们的状态。可以帮助确定任何给定环境中服务器上的 Vuser 负载。默认情况下，此图仅显示状态为 Running 的 Vuser。当然可以更改筛选条件来查看其他状态的 Vuser 情况。

X 轴表示从方案开始运行以来已用的时间，Y 轴表示方案中的 Vuser 数，如图 5-15 所示。

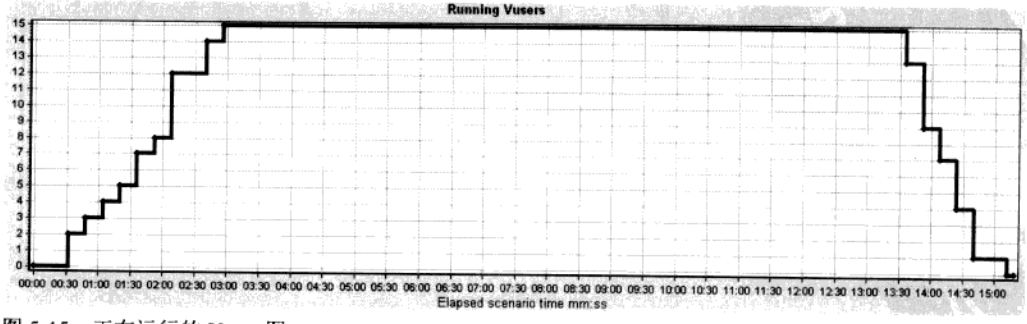


图 5-15 正在运行的 Vuser 图

Vuser 摘要图显示 Vuser 性能的摘要。使用此图可以查看运行方案成功、失败或停止的 Vuser 的情况，如图 5-16 所示。

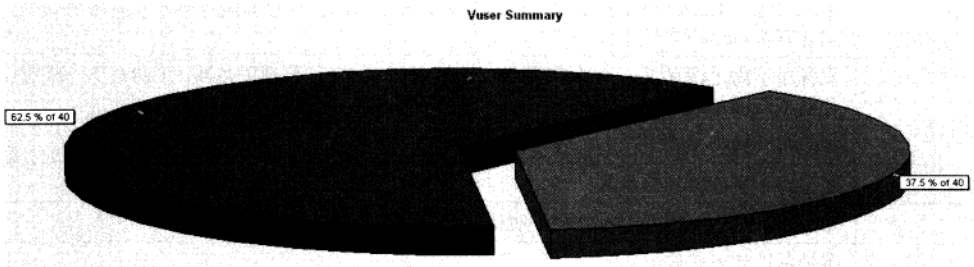


图 5-16 Vuser 摘要图

5.3.2 每秒点击数图

每秒点击数图显示在方案运行过程中 Vuser 每秒中向 Web 服务器提交的 HTTP 请求数。借助此图可以依据点击次数来评估 Vuser 产生的负载量。一般会将此图与平均事务响应时间图放到一块进行查看，观察点击数对事务性能产生的影响，如图 5-17 所示。

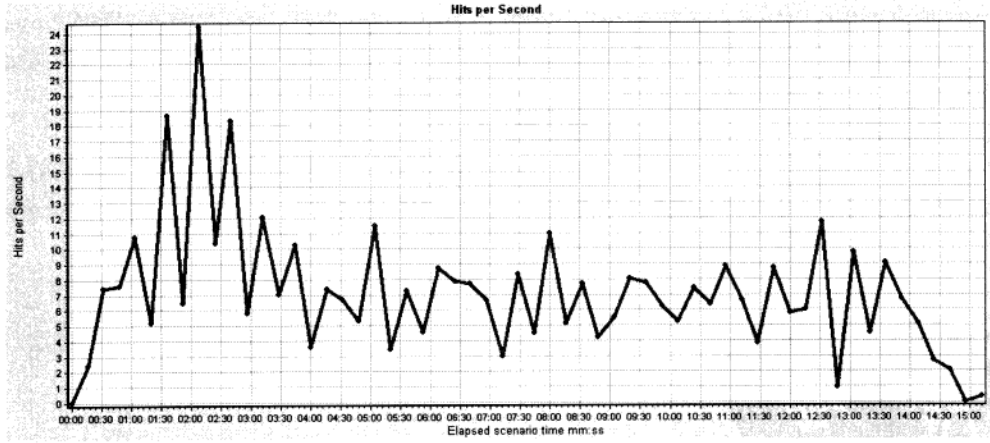


图 5-17 每秒点击数图

X 轴表示方案从开始运行以来所用的时间，Y 轴表示服务器上的点击数。通过图 5-17 能看出随着方案的运行，每秒点击数趋于稳定，说明服务器运行比较稳定。

5.3.3 平均事务响应时间图

平均事务响应时间图显示方案在运行期间执行事务所用的平均时间，如图 5-18 所示。其中 X

轴表示从方案开始运行以来已用的时间，Y 轴表示执行每个事务所用的平均时间（以秒为单位）。平均事务响应时间最直接地反应了事务的性能情况，一般会将平均事务响应时间图与 Vuser 图对照着看，来观察 Vuser 运行对事务性能的影响。

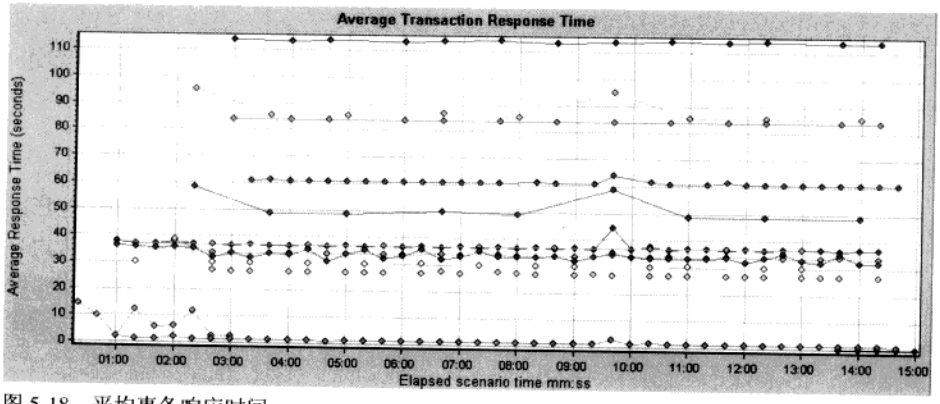


图 5-18 平均事务响应时间

在这里可以点击右键选择 Show Transaction Breakdown Tree，进一步查看子事务或者所有的事务每个页面所花费的时间。

5.3.4 吞吐量图

吞吐量图显示方案运行过程中服务器上每秒的吞吐量。吞吐量的单位为字节，表示 Vuser 在一秒时间内从服务器获得的数据量。借助此图可以依据服务器吞吐量来评估 Vuser 产生的负载量，如图 5-19 所示。可以和平均事务响应时间图对照观察，以查看吞吐量对事务性能产生的影响。

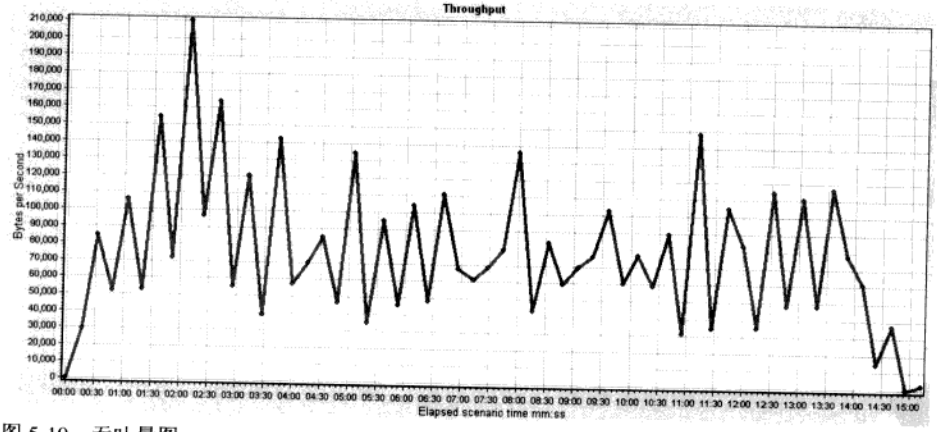


图 5-19 吞吐量图

其中 X 轴表示从方案开始运行以来已用的时间，Y 轴表示服务器的吞吐量（以字节为单位）。图 5-19 显示随着服务器的运行，吞吐量趋于平衡状态，在 80000 字节/秒附近波动。

5.4 Analysis 报告

场景运行结束后，可以在 Analysis 分析器中生成报告。Analysis 分析器提供了丰富的报告形式：HTML 格式、Word 格式和水晶报表三种格式。

5.4.1 HTML 报告

使用 Analysis 可以为方案运行创建 HTML 报告，如图 5-20 所示。保存的 HTML 报告包括在 Analysis 窗口打开的所有图和一个摘要报告。这个摘要报告与 Analysis 窗口中看到的摘要报告内容相同。而其他的视图报告数据则都来自于 Excel 文件的链接。Excel 文件保存在生成 HTML 报告时生成的 Report 文件夹下。在创建 HTML 报告时，会同时生成一个 Report 文件夹，Excel 文件链接的数据就来自这里。

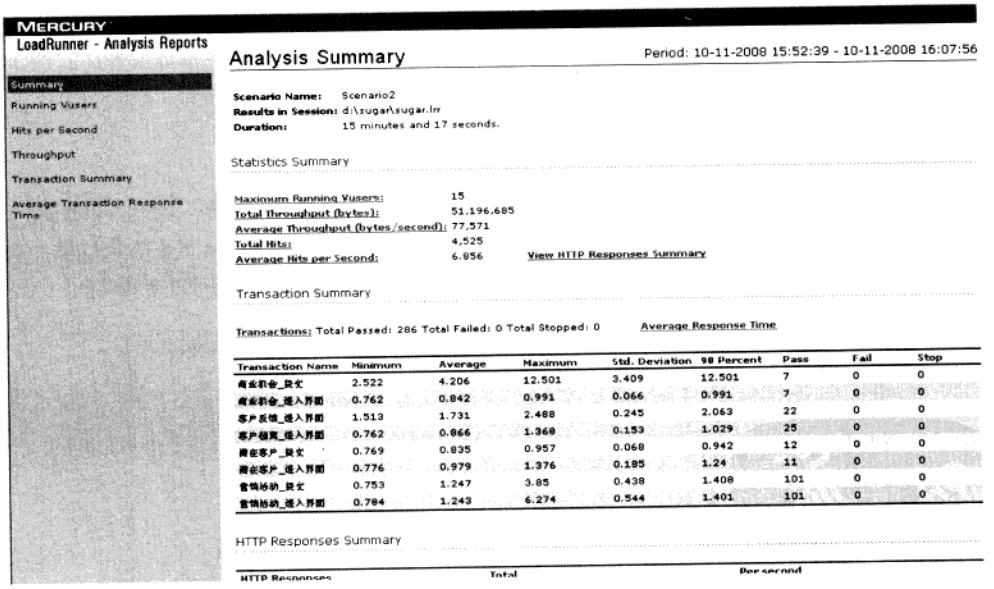


图 5-20 HTML 报告

5.4.2 Word 报告

Word 报告生成工具能够以图和表的形式自动汇总并显示测试中的重要数据。此外，它还可以显示和描述当前 Analysis 会话中的所有图。该报告还包括 LoadRunner 方案的配置概述以及一份执

行摘要，其中总结了高级注释和结论。该报告由逻辑和直觉两部分以及目录和各种附录构成。

选择 Report→Microsoft Word Report，弹出 Microsoft Word Report 对话框，如图 5-21 所示。

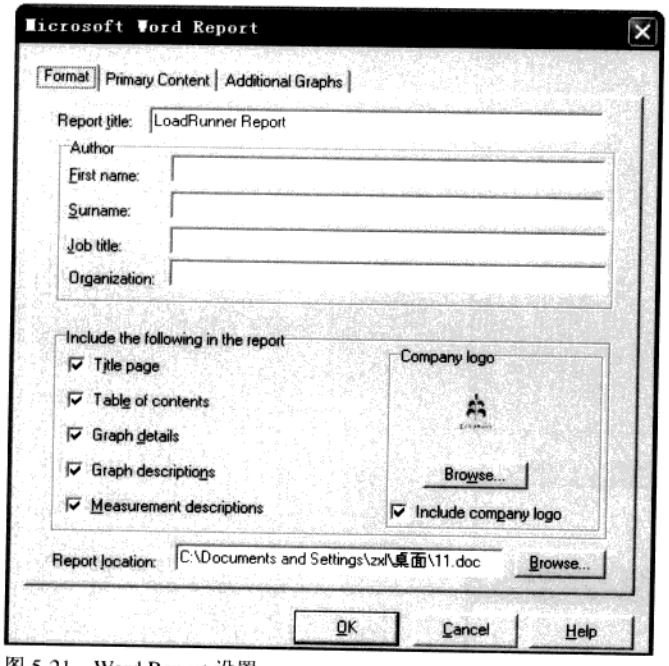


图 5-21 Word Report 设置

在该对话框看到有 Format、Primary Content 和 Additional Graphs 三个选项卡。这三个选项卡的设置内容如下：

1. Format 选项卡

标题：设置报告的标题名称。

作者信息：输入作者的相关信息，作者姓名、作者职位和组织。

报告中包含部分：选择报告中需求保存的内容，有标题页、目录、图详细信息、图描述和度量描述。

公司徽标：可以在报告中保存公司的 Logo。

报告位置：可以设置生成报告的位置。

2. Primary Content 选项卡

“Primary Content（主要内容）”选项卡用来设置报告中包括最重要性能数据的图表，还可以在报告中包括一份高级执行摘要以及方案信息，以提供该测试的概述，如图 5-22 所示。

同时也可以对摘要的内容进行编辑，点击“编辑”按钮，弹出“执行摘要”对话框，如图 5-23 所示。

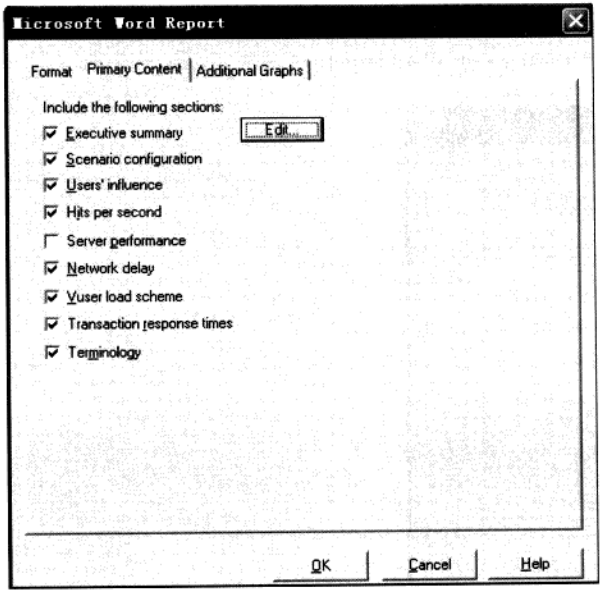


图 5-22 Primary Content 选项卡

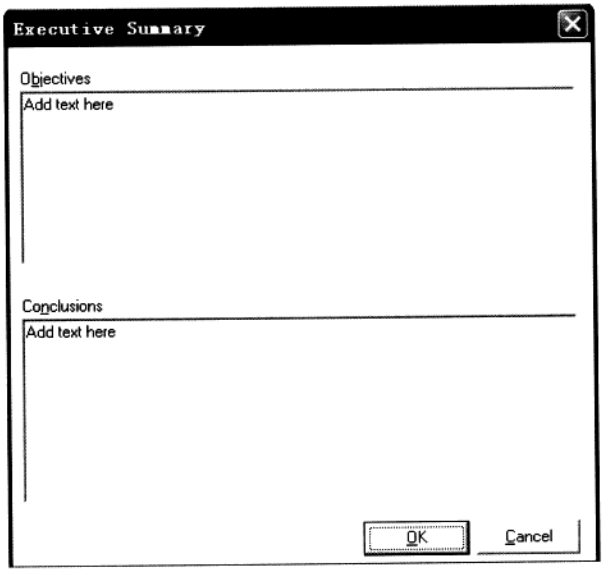


图 5-23 设置执行摘要内容

在这里可以添加目标和结论的内容。

3. Additional Graphs 选项卡

“Additional Graphs（其他图）”选项卡能够将图包括到 Word 报告中，如图 5-24 所示。在该对话框中列出当前 Analysis 会话中生成的图，还可以添加其他 LoadRunner 的图。

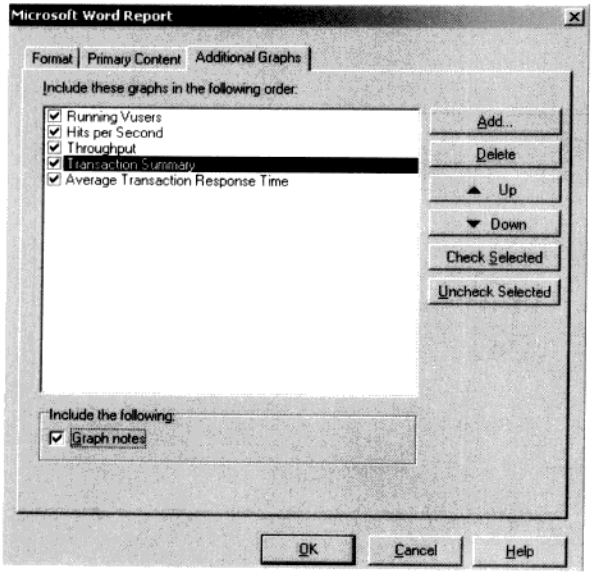


图 5-24 设置其他图选项

点击“添加”按钮，可以添加其他的分析图。

勾选“图注释（Graph notes）”选项，分析图中的注释内容也会被加载到 Word 报告中去。

5.4.3 水晶报表

水晶报表包括活动报告和性能报告两类。而活动报告又包括场景执行报告、失败事务报告和失败虚拟用户报告。性能报告又包括数据点报告、详细事务报告和事务性能报告。

1. Scenario Execution Report

选择 Reports→Crystal Report→Activity Reports→Scenario Execution Report，如图 5-25 所示。

报告中主要的内容是以场景组的形式显示所有组在场景运行期间的数据，包括虚拟用户、虚拟用户运行的主机、开始时间、运行时间、持续时间和最后运行结束后的状态。后面还有一个统计表，该统计表显示总的虚拟用户个数，以及事务运行结束后不同状态下的用户个数，即事务通过、失败、错误和停止的统计。

2. Failed Transaction Report

失败事务报告提供已完成且有失败事务的开始时间、结束时间和持续时间的详细信息。如图 5-26 所示。

Scenario Execution Report

Scenario:	Scenario2
Result:	sugar.lrr
Start Time:	10 - 十一月 - 15:52:39
End Time:	10 - 十一月 - 16:07:56
Duration:	00:15:17 (917sec)

Vusers: 40

Group: 商业机会

Vuser	*Host*	*Ready At*	*Running At*	*Duration*	*Termination Status*
Vuser1	192.168.1.8	15:52:39	15:52:39	00:00:00 (0sec)	Failed
Vuser2	192.168.1.4	15:53:25	15:53:25	00:13:28 (808sec)	Stopped
Vuser3	192.168.1.5	15:52:39	15:52:39	00:00:02 (2sec)	Failed
Vuser4	192.168.1.7	15:52:39	15:52:39	00:00:01 (1sec)	Failed

Summary

Vusers: 4

Passed: 0 ***Failed:*** 3 ***Error:*** 0 ***Stopped:*** 1

图 5-25 场景执行报告

Vuser: Vuser5

Transaction	*Start time*	*End time*	*Duration*	*Number of transactions*
客户档案_提交	15:55:19.710	15:56:11.488	00:00:51.778	1.00
Customer_Transaction	15:54:52.245	15:56:11.488	00:01:19.243	1.00
客户档案_提交	15:56:40.143	15:57:31.909	00:00:51.766	1.00
Customer_Transaction	15:56:13.364	15:57:31.909	00:01:18.545	1.00
客户档案_提交	15:57:59.982	15:58:51.788	00:00:51.806	1.00
Customer_Transaction	15:57:33.180	15:58:51.788	00:01:18.608	1.00
客户档案_提交	15:59:21.125	16:00:12.908	00:00:51.783	1.00
Customer_Transaction	15:58:53.755	16:00:12.908	00:01:19.153	1.00
客户档案_提交	16:00:40.979	16:01:32.785	00:00:51.806	1.00
Customer_Transaction	16:00:13.951	16:01:32.785	00:01:18.834	1.00
客户档案_提交	16:02:01.508	16:02:53.288	00:00:51.780	1.00
Customer_Transaction	16:01:34.556	16:02:53.288	00:01:18.732	1.00
客户档案_提交	16:03:22.007	16:04:13.961	00:00:51.954	1.00
Customer_Transaction	16:02:55.186	16:04:13.961	00:01:18.775	1.00
客户档案_提交	16:04:42.391	16:05:34.152	00:00:51.761	1.00
Customer_Transaction	16:04:15.579	16:05:34.152	00:01:18.573	1.00
客户档案_提交	16:06:02.507	16:06:54.264	00:00:51.757	1.00
Customer_Transaction	16:05:35.621	16:06:54.264	00:01:18.643	1.00

图 5-26 失败事务图

在场景执行报告或失败事务报告中发现虚拟用户执行过程中存在问题后，接着应该在失败事务报告中深入查找是哪些事务失败。当然还需要结合其他的一些信息，如应用程序日志、数据库等信息，来定位问题。

3. Failed Vusers Report

失败虚拟用户报告统计了在场景运行过程中，结果状态为“错误”、“停止”或“失败”的所有虚拟用户的详细信息，包括虚拟用户主机、准备时间、运行时间、持续时间和最后状态，如图 5-27 所示。

Failed Vusers Report

Scenario:	Scenario2
Result:	sugar.lrr
Start Time :	10 - 十一月 - 15:52:39
End Time :	10 - 十一月 - 16:07:56
Duration:	00:15:17 (917sec)

Vusers: 40

Group: 商业机会

Vuser	*Host*	*Ready At*	*Running At*	*Duration*	*Termination Status*
Vuser1	192.168.1.8	15:52:39	15:52:39	00:00:00 (0sec)	Failed
Vuser2	192.168.1.4	15:53:25	15:53:25	00:13:28 (808sec)	Stopped
Vuser3	192.168.1.5	15:52:39	15:52:39	00:00:02 (2sec)	Failed
Vuser4	192.168.1.7	15:52:39	15:52:39	00:00:01 (1sec)	Failed

Summary:

Vusers: 4

Failed: 3 ***Error:*** 0 ***Stopped:*** 1

图 5-27 虚拟用户报告

4. Data Point Report

使用 LoadRunner，可以记录自己分析的数据，也可以记录外部函数或变量的值，以及使用收集的数据创建数据点图和报告。

创建数据点会使用到 lr_user_data_point 函数，如下面一段代码。

```
for (i=0;i<100;i++) {
    measure_cpu ( );
    cpu_val=cpu_check();
    lr_user_data_point("cpu", cpu_val);
    sleep(1);
}
```

数据点报告是一种性能报告，它列出数据点的名称、值以及记录该值的时间。值将对每个组和 Vuser 都显示，如图 5-28 所示。

5. Detailed Transaction Report

详细事务报告也是性能报告的一种，提供场景运行期间每个 Vuser 执行所有事务的列表，以及每个事务执行时间的详细信息，如图 5-29 所示。详细地描述了用户执行事务的先后顺序以及所有事务执行的详细信息，主要包括以下信息。

Group: Group1

		Data Point	Time
Vuser id: 1			
	memory	19.00	13:37:16
	memory	1.00	13:37:20
	memory	9.00	13:37:32
	memory	1.00	13:37:36
	memory	1.00	13:37:40
Vuser id: 2			
	memory	6.00	13:37:05
	memory	8.00	13:37:20
	memory	9.00	13:37:32
	memory	1.00	13:37:36
	memory	1.00	13:37:40

图 5-28 数据点报告

Detailed Transaction Report (By Vuser)

Scenario:	Scenano2
Result:	sugar.lrr
Start Time:	10 - 十一月 - 08 15:52:39
End Time:	10 - 十一月 - 08 16:07:56
Duration:	00:15:17 (917sec)

Group: 客户档案

Vuser : **Vuser1**

Transaction	*Start time*	*End time*	*Duration*	*Think Time*	*Wasted Time*	*Result*	*Number of Transactions*
vuser_init_Transaction	15:52:55.914	15:52:56.0	00:00:00.1	00:00:00.000	00:00:00.000	Fail	1.00

Vuser : **Vuser2**

Transaction	*Start time*	*End time*	*Duration*	*Think Time*	*Wasted Time*	*Result*	*Number of Transactions*
vuser_init_Transaction	15:53:17.685	15:53:17.7	00:00:00.0	00:00:00.000	00:00:00.000	Fail	1.00

图 5-29 详细事务报告

- “Start time（开始时间）”：表示事务开始运行的系统时间。
- “End time（结束时间）”：表示事务运行结束时的系统时间，包括思考时间和浪费的时间。
- “Duration time（持续时间）”：表示事务运行的持续时间。
- “Think time（思考时间）”：表示事务运行期间 Vuser 思考时间。
- “Wasted time（浪费时间）”：表示既不属于事务运行时间也不属于思考时间，而是内部处理时间，这部分时间主要来自于 RTE Vuser 的时间。
- “Result（结果）”：表示事务运行结束后事务最后的状态。

6. Transaction Performance Report

事务性能报告也是性能报告的一种，显示每个 Vuser 在方案运行期间执行事务所需要的时间。报告显示了事务运行结束的状态（通过、失败或停止），并且显示了每个 Vuser 运行事务的最小时间、平均时间、最大时间和标准差，如图 5-30 所示。

Transaction Performance by Vuser Report

Scenario: Scenario2
Result: sugar.lrr
Start time: 10 - 十一月 - 0[illegible] 15:52:39
End Time: 10 - 十一月 - 0[illegible] 16:07:56
Duration: 00:15:17 (917sec)

VuserID

Transaction: Campaignsb_Transaction
Group: 营销活动经典

				Performance (sec)			
	Vuser	*Passed*	*Failed*	*Stopped*	*Min*	*Avg*	*Max*	*STD*
	Vuser1	21	0	0	36.32	36.48	37.80	.30
	Vuser2	21	0	0	36.60	37.20	44.47	1.65
	Vuser3	20	0	0	36.63	36.76	36.98	.10
	Vuser4	19	0	0	36.59	36.69	36.90	.07
Total:	**4**	**81**	**0**	**0**	**36.32**	**36.78**	**44.47**	

图 5-30 事务性能报告

第二部分

提高篇

入门篇主要介绍了性能测试基础知识和 LoadRunner 的三大组件，但介绍的仅仅是基础知识，而并未对三大组件进行深入的分析，因此，仅仅依靠现有的知识，还无法处理实际测试过程中的问题。在实际测试过程可能遇到不同的业务流程和场景模型，这样必须借助一些技巧或方法来解决实际测试过程中的问题。在提高篇中，将对 VuGen、Controller 控制台和 Analysis 分析器三大组件进行深入的分析，详细介绍在测试过程中常用的技巧与方法。

6 脚本编写

LoadRunner 录制结束后会自动生成一段脚本。这段脚本虽然很简单，但很实用，适合初学者学习。但是在真正进行项目性能测试时，只靠 LoadRunner 自动生成的脚本还是不够，很难达到业务的要求。因此，在录制脚本结束后，要对脚本进行完善，使其能达到业务模拟的要求，这样尽可能地使虚拟用户模拟时更接近用户实际使用。

本章将从以下几个方面介绍完善脚本的技巧：

- 插入检查点
- Block（块）技术
- 参数化技术
- 关联技术

6.1 检查点

在进行压力测试时，经常会有页面间数据传递的操作。如果在测试过程中传递的次数逐渐增多，页面就有可能会发生传递混乱，或者客户端与服务器端数据传输被中断或传输过程中产生了错误的数据等情况。为了判断数据传递的正确性，更重要的是为了节省人工检查的步骤和时间，LoadRunner 提供了在脚本中插入检查点的方法，在每次运行时都检查服务器返回页面的信息是否正确，这样可以大大提高测试效率。

检查点是通过检查点函数将返回值的结果反映在 Controller 的状态面板上和 Analysis 统计结果中。这个原理是基于 LoadRunner 中很多的 API 函数的返回值会改变脚本的运行结果。比如，检查点函数 web_find，如果它检查到的结果为空，它的返回值就为 LR_FAIL，这样整个结果置为 FAIL；反之，检查到的结果为成功，则 web_find 返回值是 LR_PASS，整个结果置为 PASS。

6.1.1 插入检查点

插入检查点的方法，在工作原理上是在 VuGen 中插入 Text/Image 检查点。插入检查点的步骤如下：

1）将视图模式设置为 Tree View，如图 6-1 所示。

VuGen 中包含 Tree View（树视图）和 Script View 两种视图模式。一般情况下都是使用 Script View，但在插入检查点时，一般都将视图模式切换为 Tree View，这样方便找到要插入检查点的页面，对于初学者来说这是一种很好的方式，这样可以保证插入检查点的位置正确。

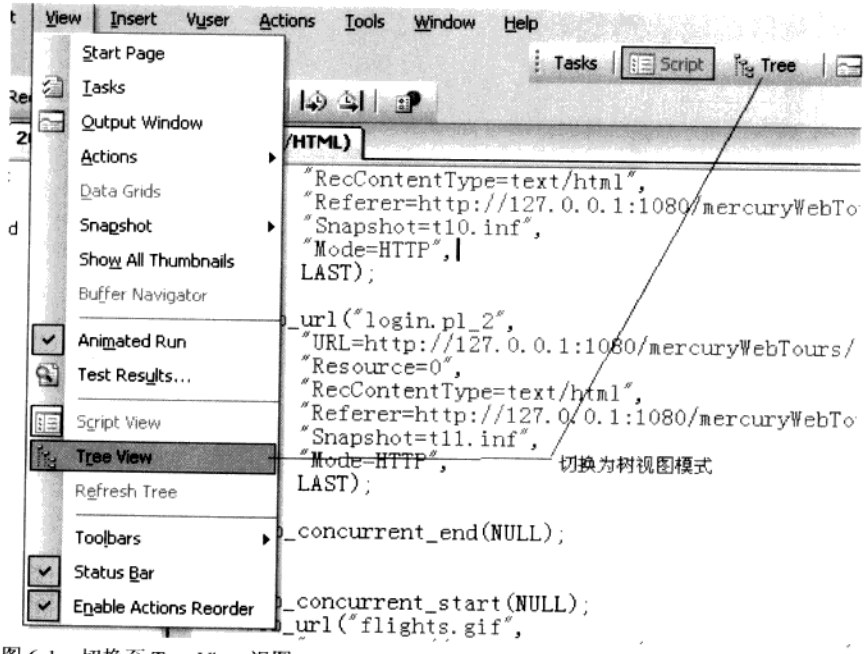


图 6-1 切换至 Tree View 视图

2）在 Tree View 中选择要插入检查点的位置。

选中要插入检查点的位置后，点击鼠标右键，选择检查点插入的方式，可以选择插在前面，也可以选择插在后面，如图 6-2 所示。

3）选择检查点类型和插入函数。

插入检查点有文本检查点（Text Check）和图片检查点（Image Check）两种，选择的检查点函数也有两个，分别为：web_find 和 web_reg_find。

插入文本检查点（Text Check），使用 web_find 检查点函数的情况，如图 6-3 所示，点击 OK 按钮后会弹出 Text Check Properties 对话框，如图 6-4 所示。

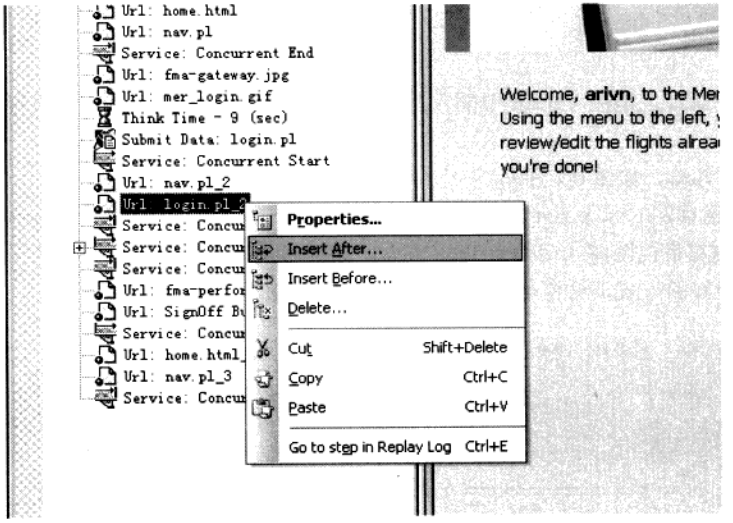


图 6-2 插入检查点位置

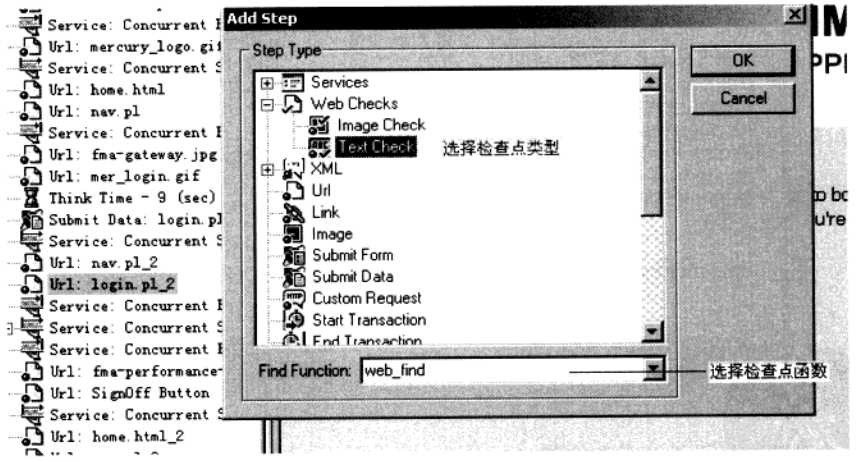


图 6-3 选择检查点类型和检查点函数

在 Specification 选项卡，设置检查参数信息，主要有以下三个属性需要设置：

- Search for：设置要检查的字符串，可以点击 对检查的内容进行参数化。
- Right of：设置右边界值，也就是设置要检查字符串右边的内容，如 How are you，如果要检查的内容为“are”，那么右边界值为“you”，也可以设置为空。
- Left of：设置左边界值，也就是设置要检查字符串左边的内容，如 How are you，如果要检查的内容为“are”，那么左边界值为“How”，也可以设置为空。

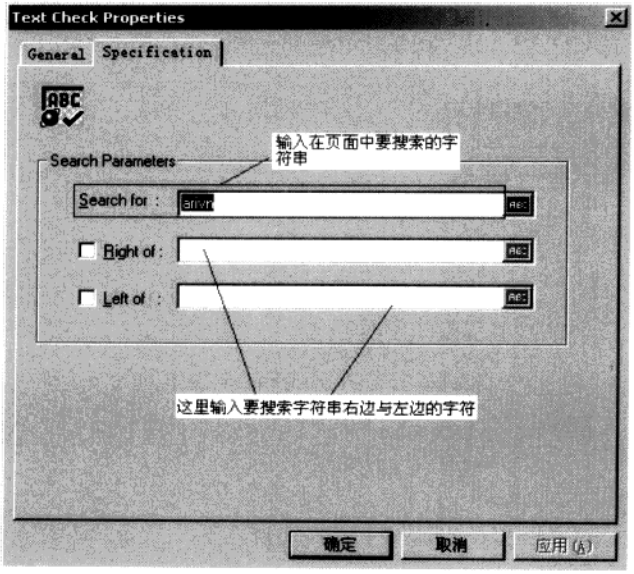


图 6-4 检查点属性设置

然后切换到 General 选项卡，如图 6-5 所示，在 Step Name 文本框中输入该操作的步骤名称，命名时名字最好能反映该操作要搜索的对象。

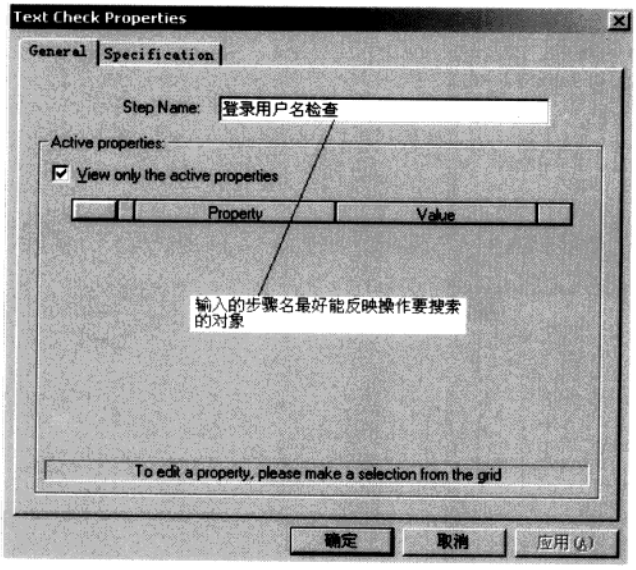


图 6-5 设置检查点的名称

插入文本检查点（Text Check），使用 web_reg_find 检查点函数的情况，如图 6-6 所示，点击 OK 按钮弹出 Find Text 对话框。

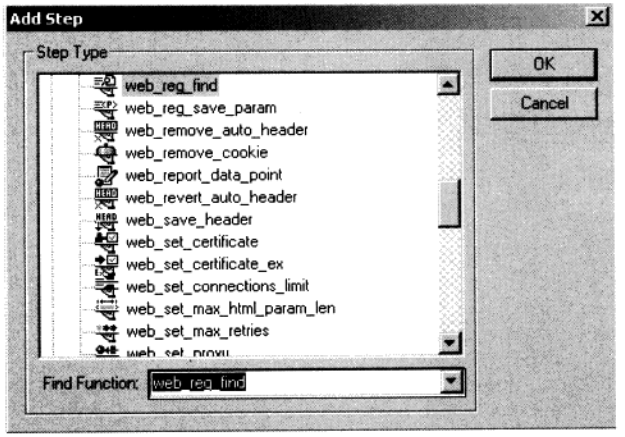


图 6-6 检查点类型与检查点函数选择

在 Find Text 对话框中设置搜索的属性配置，如图 6-7 所示。

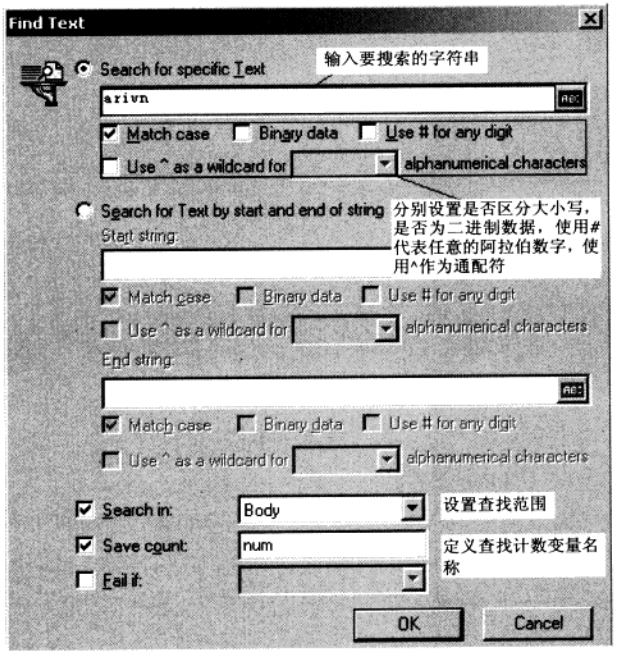


图 6-7 检查点属性设置

在 Find Text 对话框设置检查点属性时，有以下几个需要注意的地方：

1）Search for specific Text：和 web_find 检查点函数一样，同样需要对检查的字符串进行设置，也可以对检查点的内容进行参数化，但 web_reg_find 检查点函数还可以对检查的内容进行是否区分大小写、是否为二进制数据、是否使用“#”代替任意阿拉伯数字和使用“^”作为通配符的设置。

2）Search in：设置查找范围，缺省值为 Body。

3）Save count：定义查找计数器变量名称，该变量会自动统计检查内容出现的次数。

在检查点类型和函数中，选择检查点类型为 Image Check，检查点函数为 web_image_check，如图 6-8 所示。

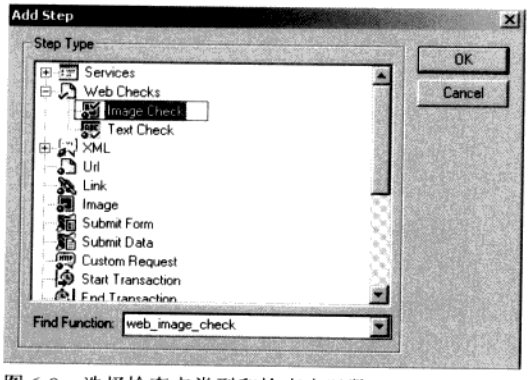


图 6-8 选择检查点类型和检查点函数

点击 OK 按钮，弹出 Image Check Properties 对话框，在该对话框中设置图片的提示信息和图片的相对路径，如图 6-9 所示。

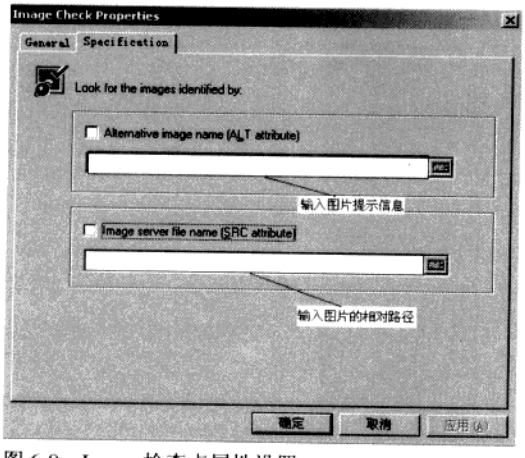


图 6-9 Image 检查点属性设置

但是如果 Web 窗体中包含有 JavaScript 脚本，那么在 Tree View 中显示可能会有问题，这时要在 General Options 中进行设置，如图 6-10 所示。

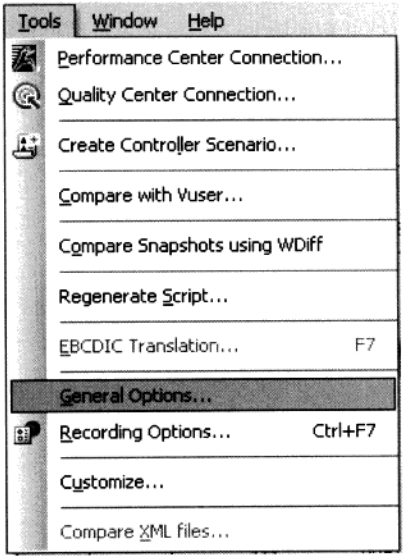


图 6-10 选择 General Options

进入 General Options 对话框，如图 6-11 所示，切换到 Correlation 选项卡，选中 Enable Scripting and Java applets on Snapshots viewer 复选框，点击 OK 按钮，设置完毕。

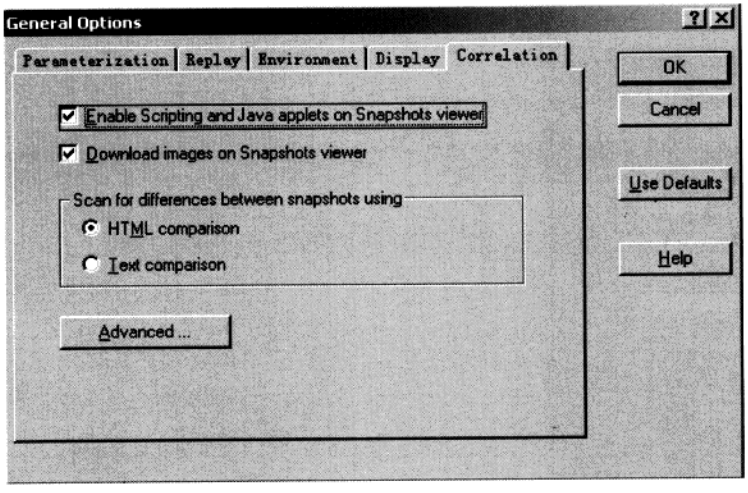


图 6-11 General Options 设置

4）参数化。检查点也可以进行参数化，因为检查点的内容可能经常变化，因此在一定的时候需要对检查点的内容进行参数化操作。在检查点属性设置对话框中点击按钮，弹出参数设置对话框，如图 6-12 所示。

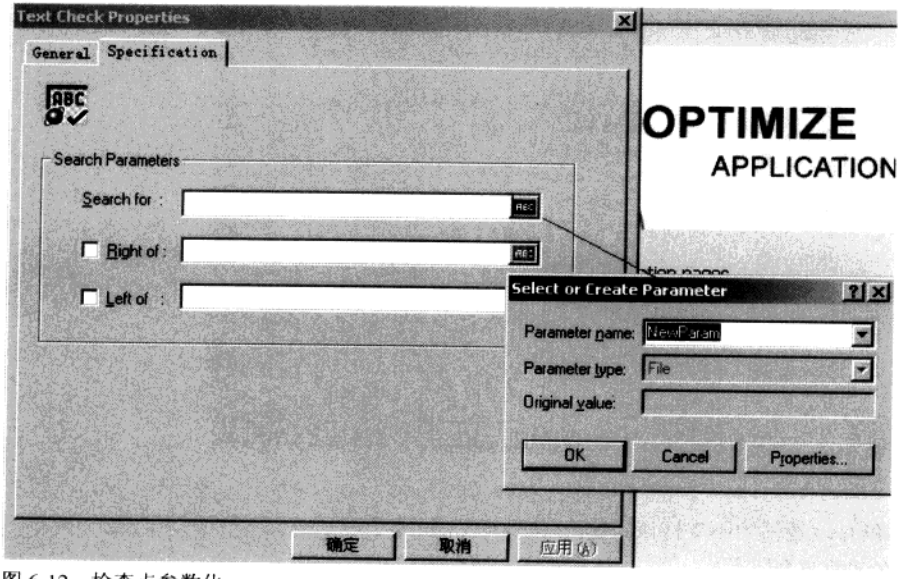


图 6-12 检查点参数化

6.1.2 检查点函数

常用的检查点函数有 web_find()和 web_reg_find()两个。

1. web_find()函数

该函数作用是在页面中查找相应的内容。常用参数含义如下：

```
web_find("Text Check",  //检查点步骤名称
        "RightOf=Go to", //定义查找字符串右边界
        "LeftOf=page",   //定义查找字符串左边界
        "What=Home",     //定义检查字符串内容
        LAST);
```

使用该函数时要注意以下几个问题：

1）该函数只能对基于 HTML 模式录制的脚本进行查找。

2）该函数必须在页面内容显示出来以后才能进行查找，所以该函数必须写在查找内容所在页面的后面。

3）必须启用内容检查选项，在 Run-time Setting→Preferences 里面，把 Enable Image and text check 复选框选中，否则不执行该查找函数，如图 6-13 所示。

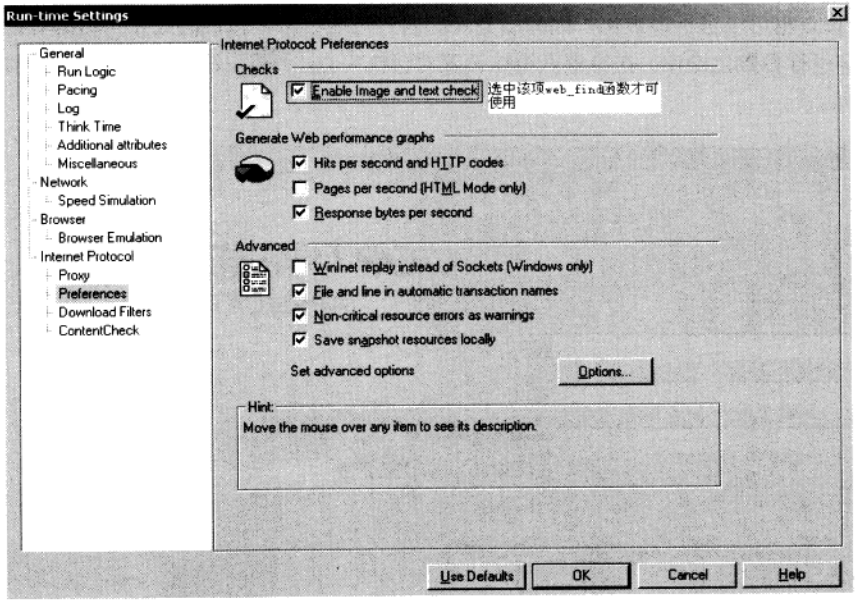


图 6-13 启用内容检查选项

4）在 VB 和 Java 语法中不支持该函数。

该函数存在下面两个缺点：

1）执行效率较低。

2）不返回查找结果情况。例如，要查看有多少个虚拟用户登录成功的情况，这个函数无法做到，必须进一步操作才能实现。

2. web_reg_find()函数

该函数是在缓存中查找相应的内容，是一个注册函数，常用参数及含义如下：

```
web_reg_find("Text=Welcome",              //定义要查找的内容
             "SaveCount=Welcome_Count",   //定义查找计数变量名
             "Search=Body",               //定义查找范围
             LAST);
```

该函数必须写在要查找内容的请求之前，一般情况下都会写在如下六个函数之前：Web_custom_request()、web_image()、web_link()、web_submit_data()、web_submit_form()、web_url()。

SaveCount 参数用来记录在缓存中内容被查找到的次数，因此在实际应用中经常会使用这个参数来统计查找成功的次数，进而来判断欲查找的内容是否真的被查找到。

下面是一个实例。

```
web_reg_find("Text=Welcome",
             "SaveCount=Welcome_Count",
             LAST);
```

```
web_submit_form("login.pl",
        "Snapshot=t2.inf",
        ITEMDATA,
        "Name=username", "Value=jojo", ENDITEM,
        "Name=password", "Value=bean", ENDITEM,
        "Name=login.x", "Value=35", ENDITEM,
        "Name=login.y", "Value=14", ENDITEM,
        LAST);
    if (atoi(lr_eval_string("{Welcome_Count}")) > 0){
        lr_output_message("Log on successful.");
        } //判断如果计数变量 Welcome_Count 值大于 0，则在日志中输出登录成功
     else{
        lr_error_message("Log on failed"); //反之则在日志中输出登录失败
        return(0);
     }
```

web_find()和 web_reg_find()虽然都是检查点函数，但两个函数还是有区别的，主要区别有以下几点：

1）两个函数类型不同，web_find 只是一个普通函数，而 web_reg_find 是一个注册函数。

2）web_find 函数使用时必须开启内容检查选项，而 web_reg_find 函数没有此限制。

3）web_find 函数录制时只能基于 HTML 模式录制的脚本中，而 web_reg_find 函数没有此限制。

4）web_find 函数是在返回的页面中进行内容查找，web_reg_find 函数是在缓存中进行查找。

5）web_reg_find 函数在执行效率上要比 web_find 函数高。

6.2 Block（块）技术

在使用 LoadRunner 时经常遇到这样一个问题，如果对不同的事务进行不同次数的循环该怎么处理？默认情况下 LR 对所有的事务都是统一执行的，即虽然有多个事务，但它们被执行的循环次数都是一样的，那么 LR 如何在一个脚本中实现不同事务不同次数的循环或不同百分比的循环呢？

案例：假设在一个脚本中，想实现注册执行 3 次，登录执行 1 次，查询执行 2 次，怎么办？录 3 个脚本？每个事务分别在脚本中复制 N 次？这样是可以解决问题，但不是最好的解决办法，LoadRunner 提供了对业务流程的处理方法，即 Block（块）技术。

首先，借用 LoadRunner 自带的订票系统，录制好这三个脚本，录制结束后，脚本如图 6-14 所示，包含三个业务：注册，登录和查询。

接着，对脚本中的三个业务的迭代次数进行设置，这里使用到的是 Block（块）技术。

1）进入菜单 Vuser→Run-time Settings，弹出 Run-time Settings 对话框，选择 General→Run Logic 选项卡，如图 6-15 所示。

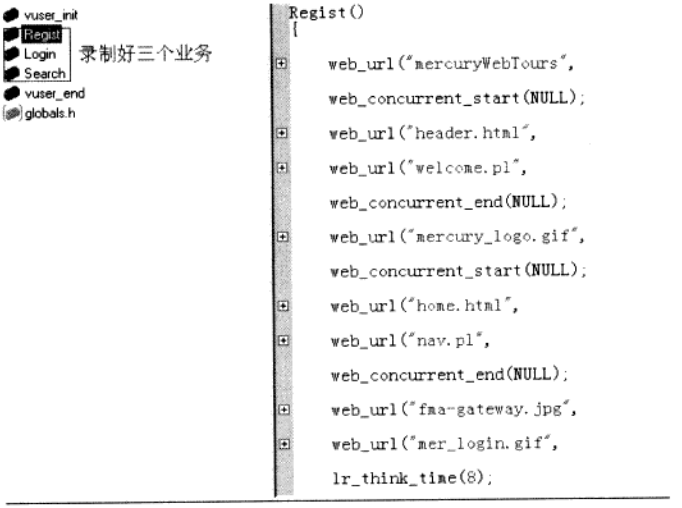


图 6-14 录制脚本

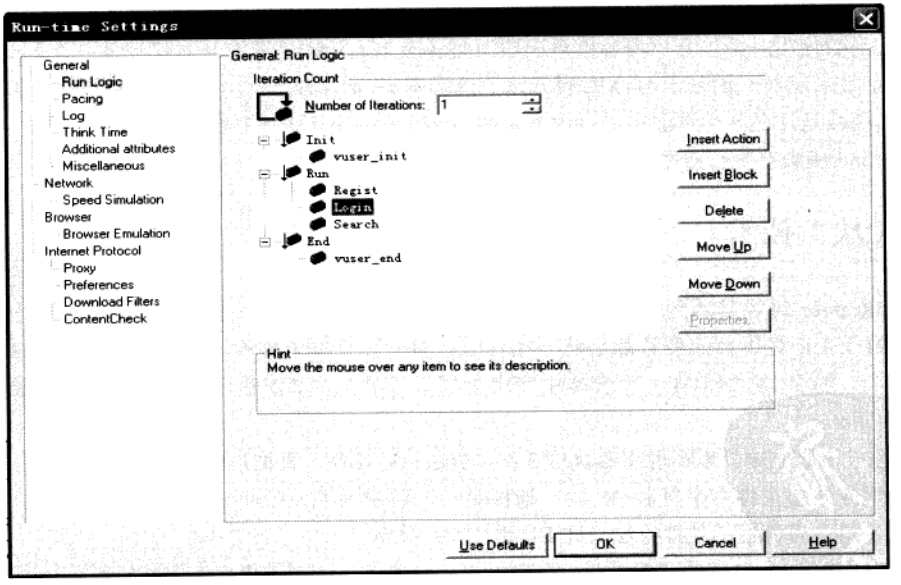


图 6-15 Run Logic 设置

2）选择 Run，插入一个 Block 块，如图 6-16 所示。

3）选择 Block0，点击 Insert Action 控钮，弹出 Select Actions 对话框，如图 6-17 所示，选中要添加的 Action，点击 OK 按钮即可。

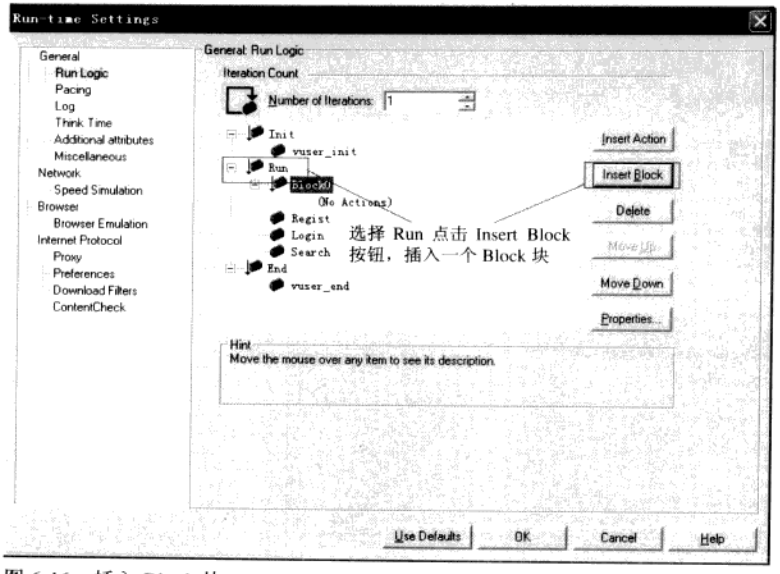


图 6-16 插入 Block 块

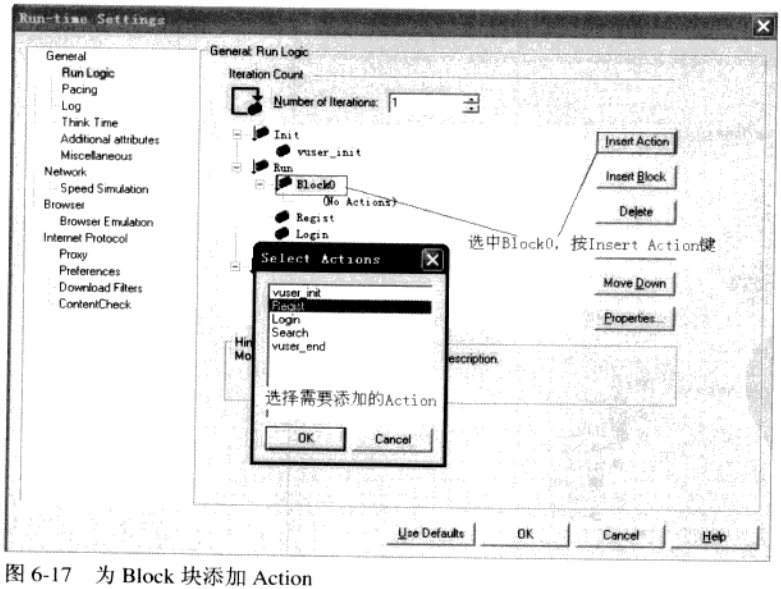


图 6-17 为 Block 块添加 Action

4）重复以上操作，再新建两个 Block 块，分别为 Block1 和 Block2，并为这两个块插入对应的 Action，如图 6-18 所示。

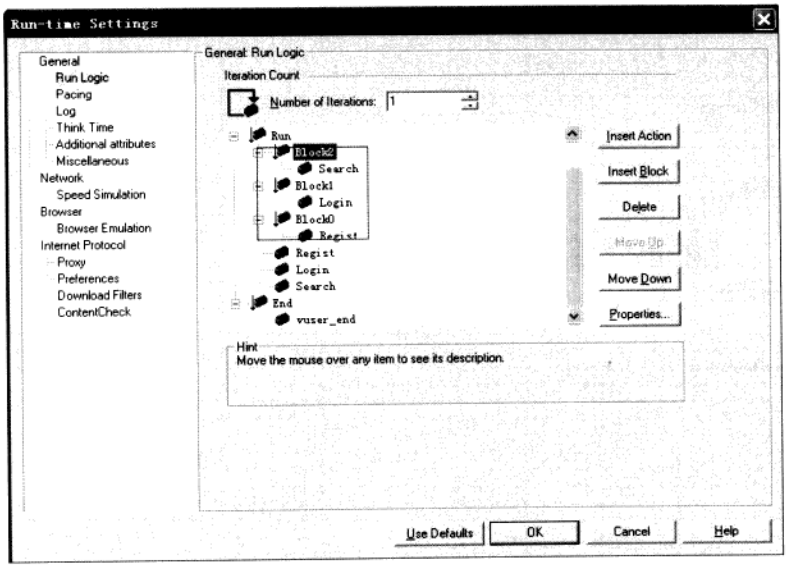


图 6-18 Block 块和 Action 插入完成

5）将 Block 外面的 Action 删除，最后得到如图 6-19 所示的 Block 块。

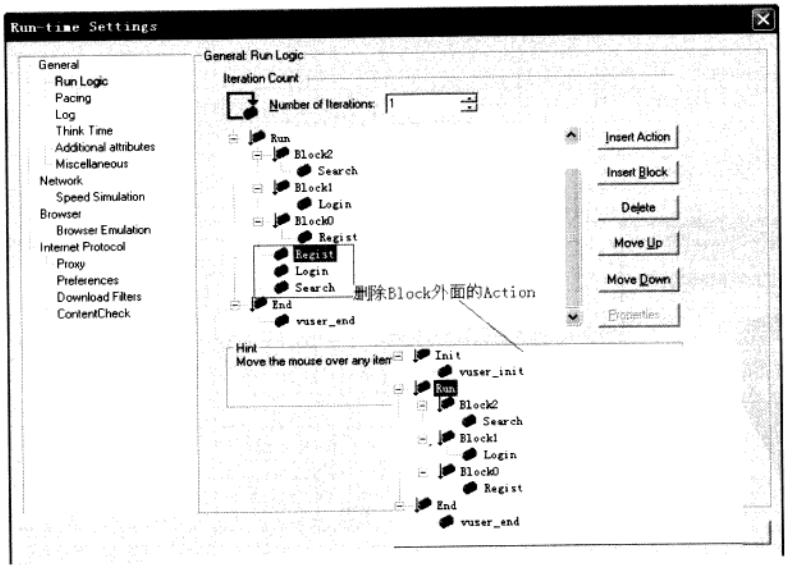


图 6-19 删除 Block 外的 Action

6）设置 Block properties。这里有两种选择：Sequential 和 Random。如果选择 Sequential，在下

面的 Iterations 中直接填入数值，那么 Block 中的 Action 都会按输入的次数执行。如果选择 Random，下面还可以设置 Block 内各 Action 执行的百分比，如图 6-20 所示。

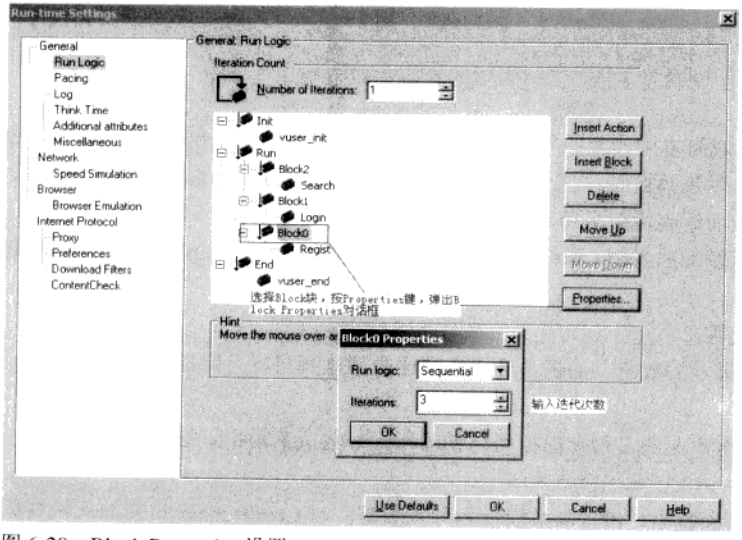


图 6-20 Block Properties 设置

按照前面的案例，只需要设置 3 个 Block，每个 Block 中分别插入一个 Action，设置执行次数分别为 3、1、2 即可，设置完成后，结果如图 6-21 所示。

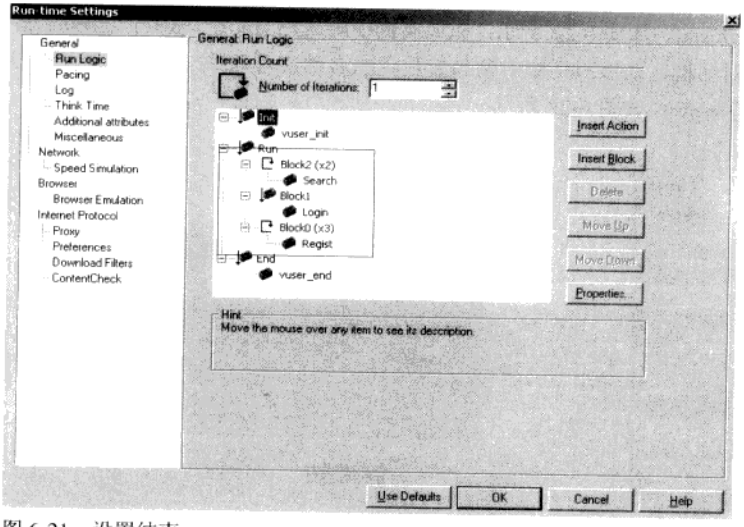


图 6-21 设置结束

整修 Block（块）的设置完成，脚本将按期望的业务模型运行。

值得注意的一点就是业务迭代的总次数=该 Block 迭代次数×Number of Iterations，如图 6-21 中 Block2 中的 Search 迭代次数为 2×1=2 次，最终 Search 这个业务只迭代了 2 次。

6.3 参数化技术

所谓脚本参数化，就是针对脚本中的某些常量，使用参数来取代。参数中包含很多数据源，数据源可以是一个文本文件也可以是数据库。当不同的 Vuser 在执行相同的脚本时，分别调用参数文件中的数据代替这些常量，从而达到模拟多用户真实使用的目的。

参数化的过程体现了数据驱动的思想，即将测试脚本与测试数据进行分离的思想。脚本体现测试流程，数据体现测试案例。

那么为什么要进行参数化呢？

1）借助参数化可以减小脚本的数量，如果不进行参数化为了达到目标可能需要拷贝并修改很多个脚本。

2）使业务更接近真实的客户业务，每个虚拟用户使用不同参数值来模拟，这样可以更好地接近客户的实际情况。

本节的主要内容包括如何参数化、设置参数属性和导入数据，以 LoadRunner 自带的飞机订票系统的注册业务过程为实例。

6.3.1 创建参数

将飞机订票系统的注册业务流程录制成脚本，录制好脚本后，选中要参数化的常量，点击鼠标右键→选择 Replace with a parameter，如图 6-22 所示，在该实例中对用户名和密码进行参数化。弹出 Select or Create Parameter 对话框，在该对话框中输入参数化的名称或者选择一个已经存在的参数名，这里将参数化名称设置为 pw，如图 6-23 所示。

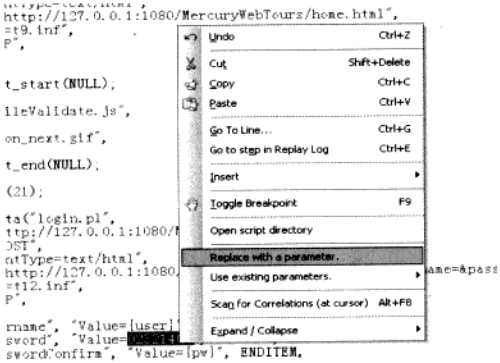


图 6-22 创建参数

图 6-23 参数化命名

需要注意的一个问题是，当参数化结束后，脚本保存的根目录下会自动生成一个参数化的文件，如图 6-24 所示。

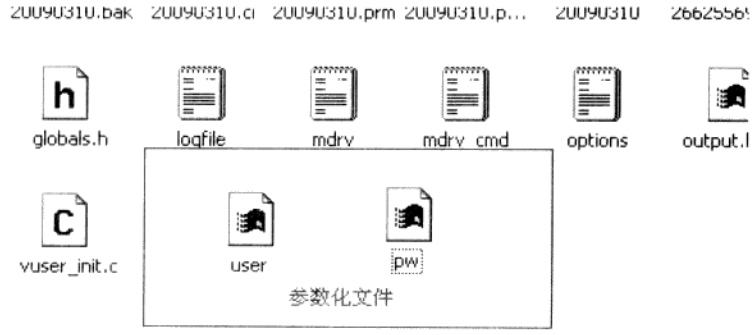


图 6-24 参数化文件

在这个实例中有两个参数化文件，在此可以对这两个参数化文件进行合并，当然并不是非合并不可，但是如果有多个参数化文件并且每个文件都占很大空间时，就需要对参数化文件进行合并了，这样不但可以节省系统资源也方便管理参数化文件，在该实例中将这两个参数文件合并成一个参数文件 parameter，在参数文件 parameter 中不同参数之间使用逗号分开，如图 6-25 所示。

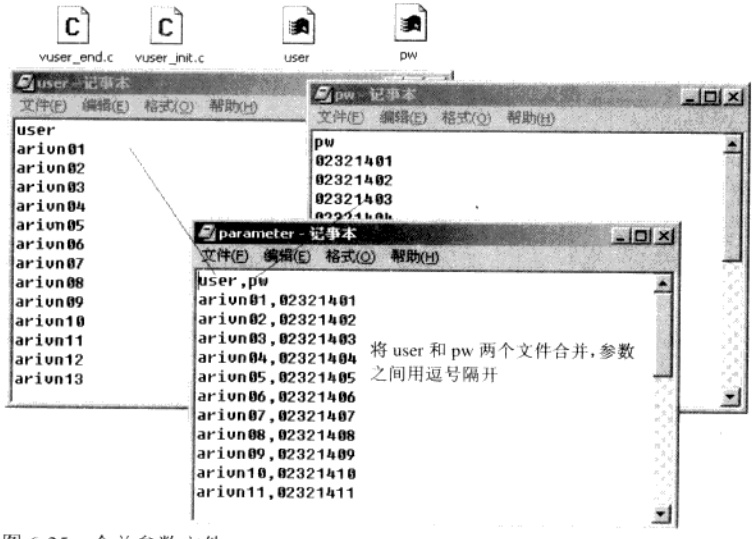


图 6-25 合并参数文件

合并好参数文件之后，可以将 user 和 pw 两个参数化文件删除，为了将参数与脚本分离，一般新建一个参数文件夹，将所有的参数文件都放到里面，如图 6-26 所示。

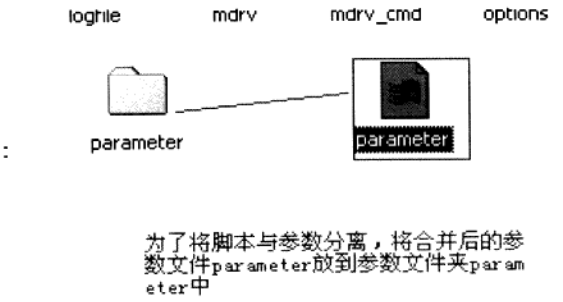


图 6-26 脚本与参数分离

创建参数完成后，需要对参数类型属性进行设置。

6.3.2 参数类型属性

参数创建好之后，需要对这些参数的类型进行设置，主要有以下几种参数类型：

1）Date/Time（日期/时间）参数类型：Date/Time 类型用当前的日期/时间替换参数。要指定日期/时间的格式，可以从菜单列表中选择，或者指定实际需要的格式，该格式应该与脚本中录制的日期/时间格式相对应。还可以点击该对话框中相应的按钮对格式进行添加、删除、还原等操作。

2）Group Name（组名）参数类型：用 Vuser 组的名称替换参数。创建方案时，要指定 Vuser 组的名称，否则运行 VuGen 的脚本时，组名始终为“无”。

3）Iteration Number（迭代编号）参数类型：用当前的迭代编号替换参数。

4）Load Generator Name（负载发生器名）参数类型：用 Vuser 脚本的负载发生器名替换参数。负载发生器是运行 Vuser 的计算机。

5）Random Number（随机编号）参数类型：用一个随机生成的整数替换参数，可以通过指定最小和最大值，设置随机编号的范围。

6）Unique Number（唯一编号）参数类型：用一个唯一编号替换参数。Block size（块大小）指明分配给每个 Vuser 的编号块的大小。每个 Vuser 都从其范围的下限（start）开始，在每次迭代时递增该参数值。

7）Vuser ID 参数类型：LoadRunner 使用该虚拟用户的 ID 来代替参数值，该 ID 由 Controller 来控制。在 VuGen 中运行脚本时，VuGen 将会是-1。

8）File 参数类型：可以在参数属性中编辑参数文件，也可以直接选择已编辑好的参数文件，还可以从现成的数据库中提取，这是最常用的一种参数化方式。

6.3.3 数据文件

参数类型设置完成后，需要设置参数的数据源，根据不同的业务需求需要不同的数据，这些数据可以是来自真实的历史数据也可以是为了测试而构建的数据，同时对参数中的数据如何调用也需要设置。

1. Browse 属性

该设置项用来选择参数文件的路径，需要注意的一个问题是，一般在做参数化的时候没有单独把参数文件放到一个文件夹下，便无需修改，但是如果将参数化文件合并成一个文件并放到一个专门管理参数的文件夹下，就要选择参数的路径，否则无法读到参数文件中的参数，具体如图 6-27 所示。

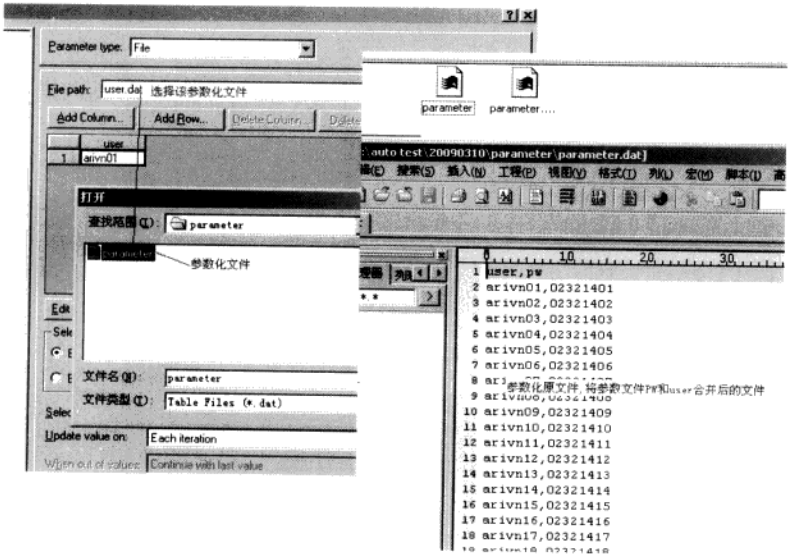


图 6-27 选择参数化文件

选择好参数文件后，参数文件中所有的数据将会被显示出来，如图 6-28 所示。

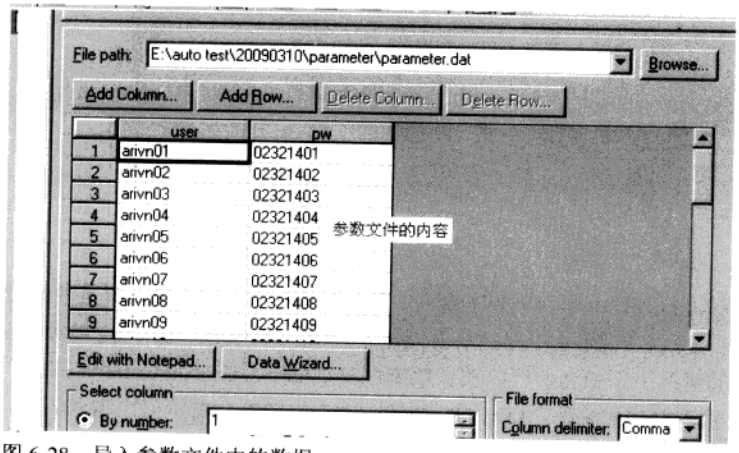


图 6-28 导入参数文件中的数据

这样设置同样有一个问题，如果把这个脚本拷贝到别的机器上运行，或者是脚本的路径发生了变化，这个路径就是错误的路径，也就是现在的脚本可移植性不好，当更换路径后，运行时一定会出错，因为这里写的是绝对路径，如果换到其他的一个盘或机器，运行就报错了，那么怎么解决呢？这里采用相对路径来解决这个问题，将 Browse 设置为相对路径，将脚本的根目录使用“.”来代替，如图 6-29 所示，这样就不会出错了。

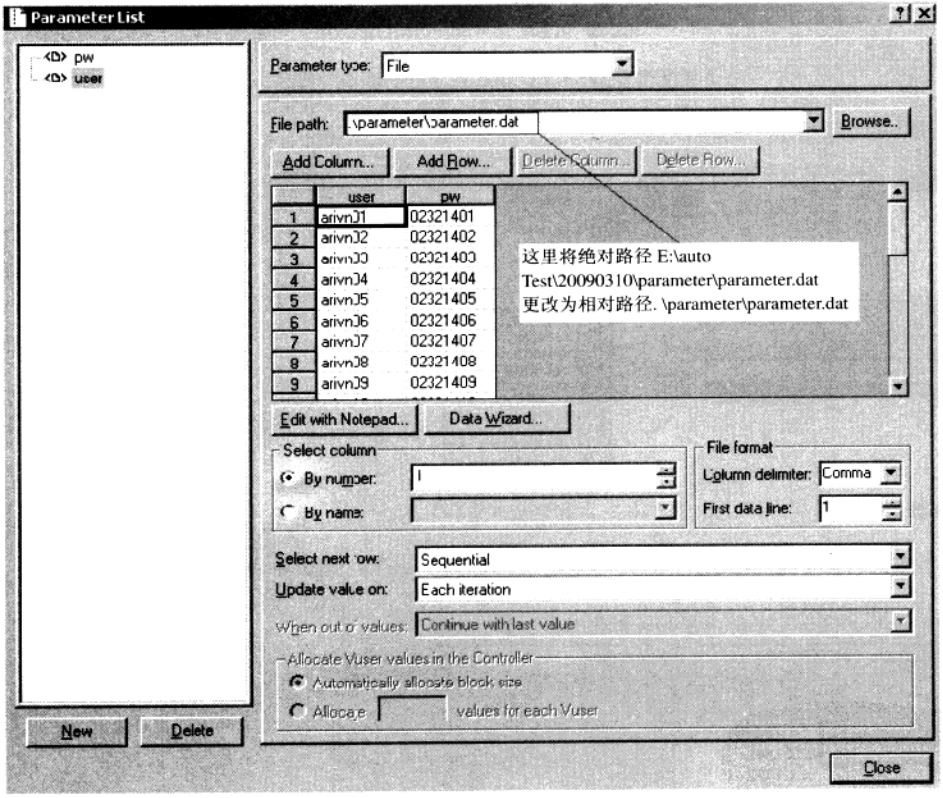


图 6-29 绝对路径更改为相对路径

2. Edit With Notepad 设置

点击 Edit With Notepad 按钮，打开记事本，记事本内容中第一行是参数名称，第二行是参数的初始值。参数之间使用逗号隔开。可以在记事本中对参数值进行修改或添加、删除，如图 6-30 所示。

如果要在没有启动记事本的情况下添加列，可以在参数属性对话框中点击 Add Column 按钮，弹出 Add new column 对话框，输入新列的名称，点击 OK 按钮，脚本生成器会将该列添加到表中，并显示该列的初始值，如图 6-31 所示。

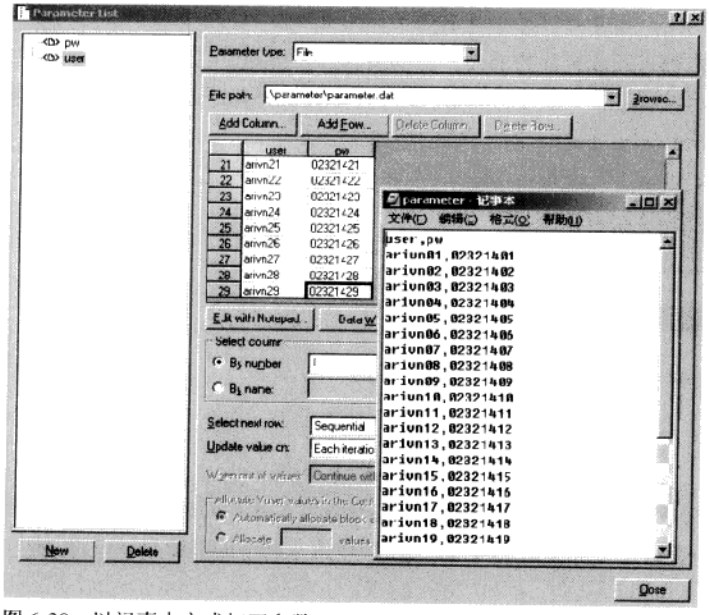


图 6-30 以记事本方式打开参数

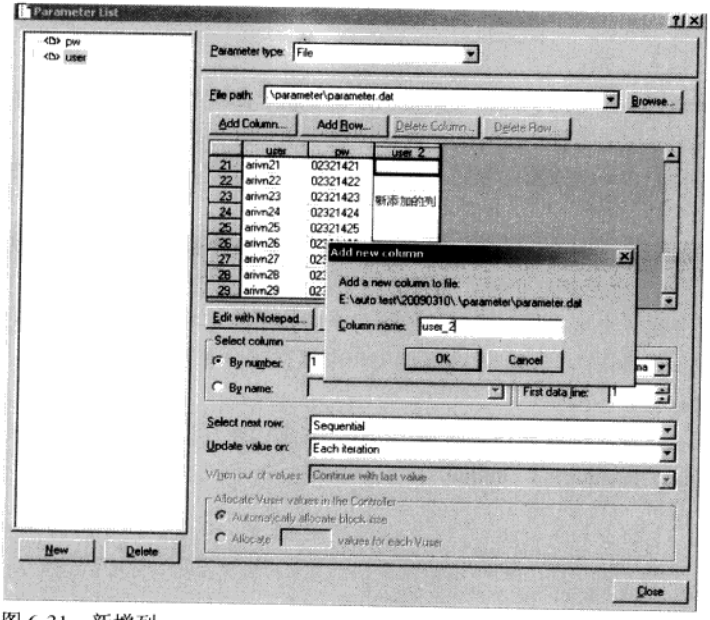


图 6-31 新增列

3. Select column 设置

指明参数选择的列。这里有两种方式选择参数的列，可以按列号来选择也可以按列名来选择。列号是包含所需要数据的列的索引，列名显示在每列的第一行，一般情况下为了避免出错，都选择按列名来选择参数数据的列，如图 6-32 所示。

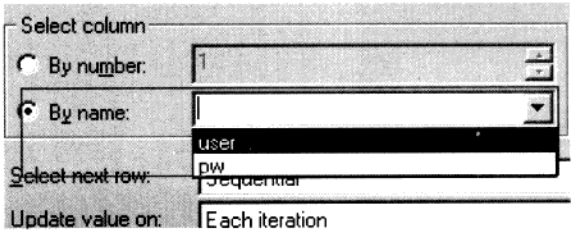


图 6-32 按列名选择

4. Column delimiter 设置

选择列分隔符，一般情况下都是默认选择逗号作为分隔符，但其实这里也可以指定逗号、空格符等进行分隔，列分隔符指的是当参数化文件中出现多列参数时(如该实例中的参数有 user 和 pw)，参数与参数之间如何隔开。

5. File data line 设置

在脚本执行时选择第几行数据开始使用。如果选择从列标题后的第一行参数开始执行的话，就在 File data line 中输入 1。

6. Select next row 设置

值的注意的是所有的 Select next row 属性选择是针对虚拟用户来说的，也就是这里的策略是针对 Controller 设置的，在调试脚本的过程中是看不出来的，其决定虚拟用户选择参数的过程。

- 顺序（Sequential）：虚拟用户 Vuser 按照行顺序读取参数文件中的数据，如果参数文件中的数据都执行了一遍，则返回到第一行，继续执行。
- 随机（Random）：每个 Vuser 从表中随机地读参数数据，假设有 50 个数据，那么随机数将在 1～50 之间随机取一个，然后把这个数作为行号，去读取相应行的参数数据。
- 唯一（Unique）：该方法分配一个唯一的有顺序的值给每个虚拟用户作为参数。例如，现在有 50 行数据，那么只能取 50 次，当第 51 个虚拟用户来执行时，就无法分配数据了，这时候 LoadRunner 就会报错。
- Same link as ＊＊＊：与某个已定义好的参数取同一行值。注意：该方法要求至少其中的一个参数必须是 Sequential、Random 或 Unique。

如下面的数据表有 2 列：

User	PW
T01	0001
T02	0002

T03 0003

将参数 PW“Select next row”设置为“Same Line as user”。当 User 选择 T01 时，那么参数 PW（密码）只能选择 0001。

7. Update value on 设置

设置脚本迭代过程中取值的策略，其结果可以在代码调试的日志中体现。

如图 6-33 所示对登录用户名和密码进行参数化。

```
web_submit_data("login.pl",
    "Action=http://127.0.0.1:1080/MercuryWebTours/login.pl",
    "Method=POST",
    "RecContentType=text/html",
    "Referer=http://127.0.0.1:1080/MercuryWebTours/login.pl?username=&passwor
    "Snapshot=t12.inf",
    "Mode=HTTP",
    ITEMDATA,
    "Name=username", "Value={user}", ENDITEM,
    "Name=password", "Value={pw}", ENDITEM,
    "Name=passwordConfirm", "Value={pw}", ENDITEM,
    "Name=firstName", "Value={user}", ENDITEM,
    "Name=lastName", "Value=", ENDITEM,
    "Name=address1", "Value=", ENDITEM,
    "Name=address2", "Value=", ENDITEM,
    "Name=register.x", "Value=74", ENDITEM,
    "Name=register.y", "Value=10", ENDITEM,
    LAST);
```

密码使用两次

用户名使用两次

图 6-33 参数定义

- Each iteration：脚本每迭代一次都访问数据表中的下一个值，也就是说在同一次迭代的过程中，不管同一个参数出现多少次都只使用这一个参数，如图 6-34 所示是参数第一次迭代的结果。

```
(92): Registering web_url("profileValidate.js") was successful          [MsgId: MM
(100): Registering web_url("button_next.gif") was successful    [MsgId: MMSG-2
(108): web_concurrent_end was successful, 12347 body bytes, 345 header bytes
(112): Notify: Parameter Substitution: parameter "user" = "arivn01"
(112): Notify: Next row for parameter pw = 1 [table = pw].
(112): Notify: Parameter Substitution: parameter "pw" = "02321401"
(112): Notify: Next row for parameter pw = 1 [table = pw].
(112): Notify: Parameter Substitution: parameter "pw" = "02321401"
(112): Notify: Parameter Substitution: parameter "user" = "arivn01"
(112): web_submit_data("login.pl") was successful, 2640 body bytes, 226 header
(131): web_url("fma-performance-center.jpg") was successful, 27000 body bytes,
ction Action.
Iteration 1.
Iteration 2.
```

用户名和密码在同一次迭代中，虽然出现两次，但都使用同一个数据

图 6-34 Each iteration 迭代结果

- Each occurrence：在每次迭代的过程中，参数的值都更新，即使在同一次迭代过程，如果某个参数使用了多次，其选择的值也会更新而并不会使用相同的参数值，如图 6-35 所示。
- Once：在同一个 Vuser 中一直取同一个参数，表中其他的数据不参与迭代的过程，如图 6-36 所示。

```
; web_url( profilevaluate.js ) was successful          [MsgId: MMSG-26390]
ig web_url("button_next.gif") was successful     [MsgId: MMSG-26390]
rent_end was successful, 12347 body bytes, 345 header bytes     [MsgId: MMSG-26386]
:tting new value for parameter 'user': table = '.\parameter\parameter.dat' column = '0'
ırameter Substitution: parameter "user" =  "arivn01"
:tting new value for parameter 'pw': table = '.\parameter\parameter.dat' column = '1' r
ırameter Substitution: parameter "pw" =  "02321401"
:xt row for parameter pw = 2 [table  = pw].
:tting new value for parameter 'pw': table = '.\parameter\parameter.dat' column = '1' r
ırameter Substitution: parameter "pw" =  "02321402"
:xt row for parameter user = 2 [table  = user].
:tting new value for parameter 'user': table = '.\parameter\parameter.dat' column = '0'
ırameter Substitution: parameter "user" =  "arivn02"
:_data("login.pl") was successful, 2558 body bytes, 226 header bytes     [MsgId: MMSG-26
:ma-performance-center.jpg") was successful, 27000 body bytes, 167 header bytes     [Ms
```

在同一次迭代过程中，用户名和密码使用了两次，取值也发现更新

图 6-35　Each occurrence 迭代

```
c(112): Notify: Parameter Substitution: parameter "user" =  "arivn01"
c(112): Notify: Next row for parameter pw = 1 [table  = pw].
c(112): Notify: Getting new value for parameter 'pw': table = '.\parameter\parameter.da
c(112): Notify: Parameter Substitution: parameter "pw" =  "02321401"
c(112): Notify: Next row for parameter pw = 1 [table  = pw].
c(112): Notify: Next row for parameter pw = 1 [table  = pw].
c(112): Notify: Getting new value for parameter 'pw': table = '.\parameter\parameter.da
c(112): Notify: Parameter Substitution: parameter "pw" =  "02321401"
c(112): Notify: Parameter Substitution: parameter "user" =  "arivn01"
c(100): Registering web_url( button_next.gif ) was successful     [MsgId: MMS
c(108): web_concurrent_end was successful, 12347 body bytes, 345 header byte
c(112): Notify: Parameter Substitution: parameter "user" =  "arivn01"
c(112): Notify: Next row for parameter pw = 1 [table  = pw].
c(112): Notify: Next row for parameter pw = 1 [table  = pw].
c(112): Notify: Getting new value for parameter 'pw': table = '.\parameter\parameter.da
c(112): Notify: Parameter Substitution: parameter "pw" =  "02321401"
c(112): Notify: Next row for parameter pw = 1 [table  = pw].
c(112): Notify: Next row for parameter pw = 1 [table  = pw].
c(112): Notify: Getting new value for parameter 'pw': table = '.\parameter\parameter.da
c(112): Notify: Parameter Substitution: parameter "pw" =  "02321401"
c(112): Notify: Parameter Substitution: parameter "user" =  "arivn01"
```

第一次迭代与第二次迭代使用的数据是同一个数据，表中的数据并没有参与迭代

图 6-36　Once 迭代

6.3.4　导入数据

LoadRunner 允许利用参数化从已经存在的数据库中导入数据。LoadRunner 提供以下两种方式：

- 使用 Microsoft Query（要求在系统上先安装 Microsoft Query）。
- 指定数据库连接字符串和 SQL 语句。

如果在业务中希望参数是来自实际的数据，通过以上两种方式就可以解决。脚本生成器在从数据库中导入数据的过程中提供了一个向导。在向导中，指明如何导入数据——通过 Microsoft Query 创建查询语句或直接使用 SQL 查询语句。在数据导入后，LoadRunner 都以.dat 为后缀并作为正规的参数文件保存。要开始导入数据库中的数据时，在参数属性对话框中点击 Data Wizard 按钮，则打开数据库查询向导。

1. Microsoft Query 创建查询

在数据查询向导中选择 Create query using Microsoft Query，如图 6-37 所示。如果没有安装 Microsoft Query，LoadRunner 会提示这个功能不能用，在进行该操作之前，必须在 Microsoft Office 中先安装好 Microsoft Query。

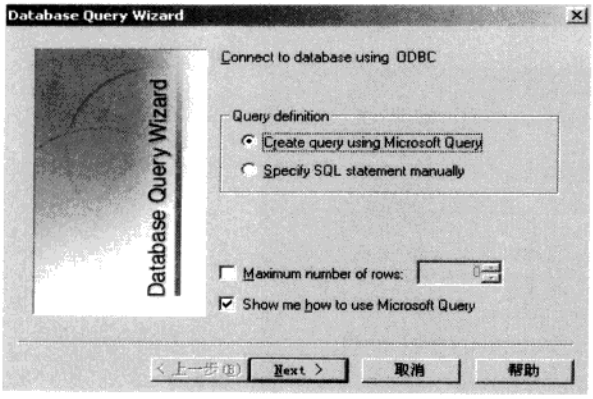


图 6-37 通过 Microsoft Query 创建查询

选择需要的数据源，如图 6-38 所示。

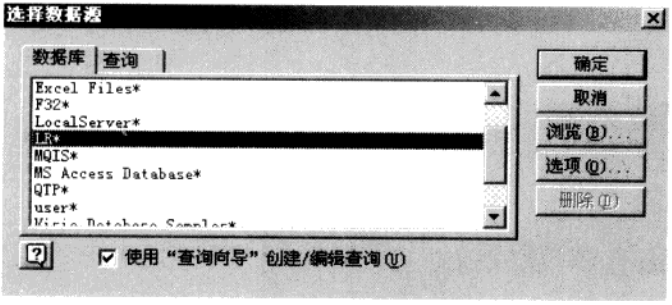


图 6-38 选择数据源

选择可用的表和列，如图 6-39 所示。

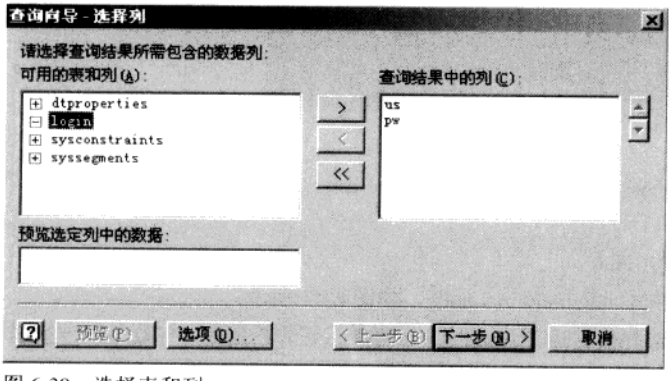


图 6-39 选择表和列

设置筛选数据条件，如图 6-40 所示。

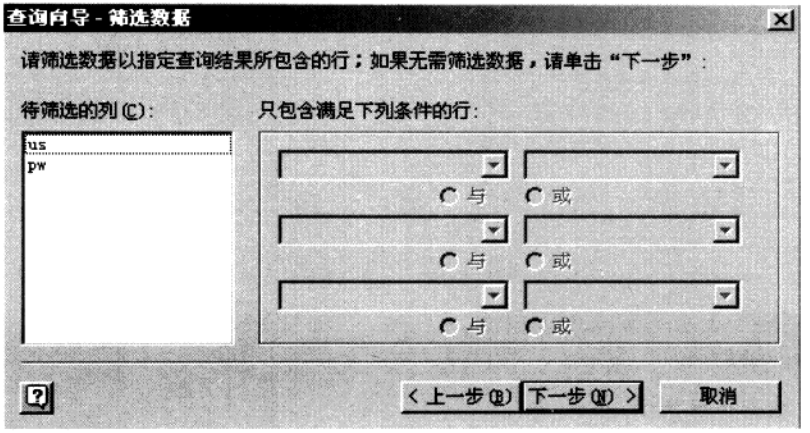


图 6-40 筛选数据

设置排序顺序，如图 6-41 所示。

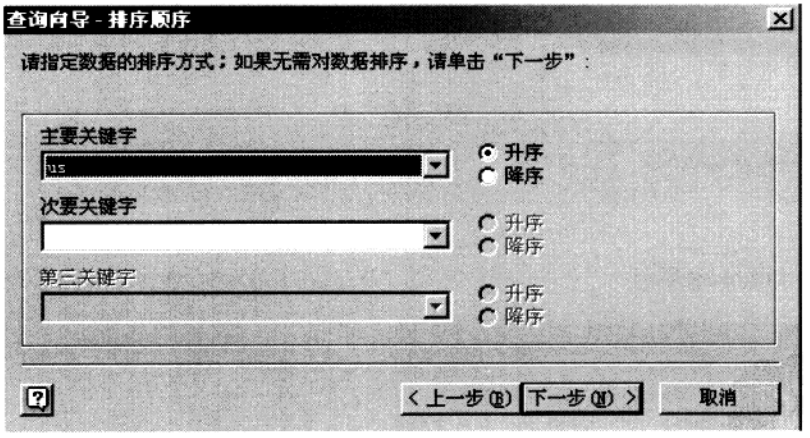


图 6-41 设置排序顺序

在数据的导入完成后，选择 Exit and return to Mercury Virtual User Generator，然后点击“完成”按钮，如图 6-42 所示。

2. 指定数据库连接字符串和 SQL 语句

在数据查询向导中选择 Specify SQL statement manually，如图 6-43 所示。

点击 Create 按钮，在弹出的“选择数据源”对话框中，选择需要的数据源，创建数据库连接字符串，如图 6-44 所示。

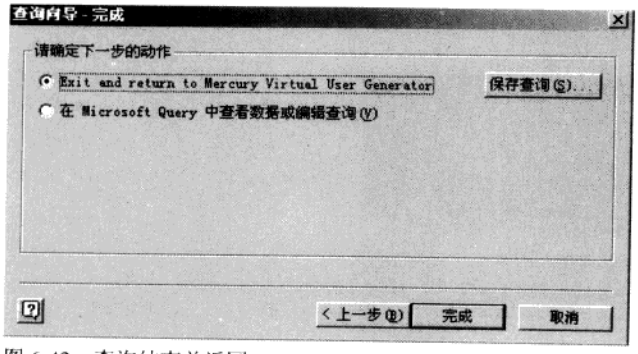


图 6-42 查询结束并返回

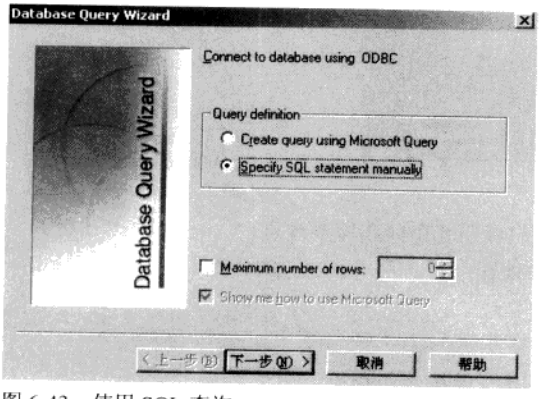


图 6-43 使用 SQL 查询

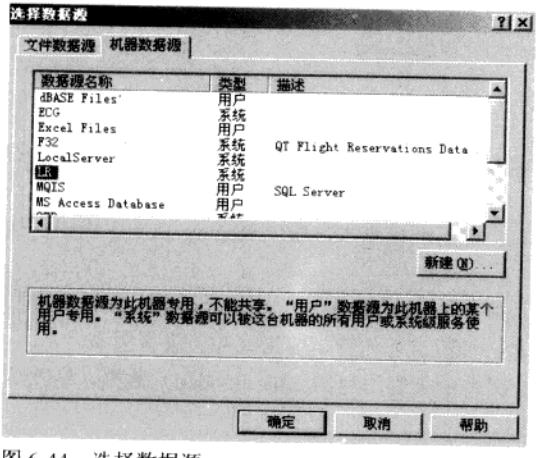


图 6-44 选择数据源

在 SQL statement 文本框中，输入 SQL 查询语句，如图 6-45 所示。

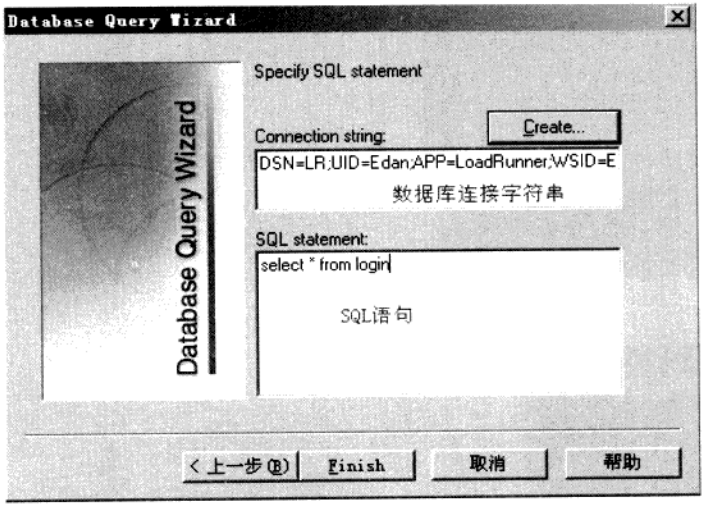


图 6-45　数据库连接字符串与 SQL 语句

以上是参数化的整个过程，但参数化过程中有以下几个问题需要注意：

- 参数化文件尽可能少，因为参数是放在内存中的，占用了内存的资源。
- 参数化文件与脚本分离。
- 参数文件的路径应该设置为相对路径。
- 为了使参数更具有真实性，参数应该从历史数据库中获得。
- 参数类型的选择。
- 参数的数据由业务决定。

6.4　关联技术

首先准备一个场景，录制这样一个脚本，登录 LoadRunner 自带的机票预订系统，检查登录的用户名是否正确，如图 6-46 所示。

录制结束之后，为登录的用户名做一个检查点，然后回放脚本，发现 Replay Log 中报错，如图 6-47 所示。

这时首先可能想到的第一个原因就是关联。关联是 LoadRunner 中一个很重要的应用，对于初学者来说也是最容易犯错的地方，但是很遗憾的是，并没有任何特定的错误与关联有关系。

那么什么叫关联呢？关联（Correlation）是把脚本中某些写死的（hard-coded）数据，转变成取自服务器所送的、动态的、每次都不一样的数据。

常用的关联技术有三种。

图 6-46 录制场景

```
Action.c(65): web_url("tma-gateway.jpg") was successful, 46063 body bytes, 167 header bytes      [MsgId:
Action.c(73): web_url("mer_login.gif") was successful, 679 body bytes, 164 header bytes      [MsgId: MMS
Action.c(83): web_submit_data("login.pl") was successful, 748 body bytes, 225 header bytes      [MsgId:
Action.c(99): web_concurrent_start was successful    [MsgId: MMSG-26392]
Action.c(101): Registering web_url("nav.pl_2") was successful    [MsgId: MMSG-26390]
Action.c(110): Registering web_reg_find was successful      [MsgId: MMSG-26390]
Action.c(113): Registering web_url("login.pl_2") was successful      [MsgId: MMSG-26390]
Action.c(122): Error -26366: "Text=arivn" not found for web_reg_find    [MsgId: MERR-26366]
Action.c(122): web_concurrent_end highest severity level was "ERROR", 2328 body bytes, 478 header bytes
Ending action Action.
Ending Vuser...
Starting action vuser_end.
Ending action vuser_end.
Vuser Terminated.
```

图 6-47 Replay Log 报错

6.4.1 录制中关联

VuGen 内建自动关联引擎（auto-correlation engine），可以自动找到需要关联的值，并且自动关联函数建立关联。

1. 建立规则

如果在录制之前已经知道关联规则，那么可以先建立一个规则，再进行录制，这样在录制过程中会自动关联。关联规则最重要是指定两个边界，即被关联量的左边界和右边界。对于关联的规则有两种。

（1）内建关联规则（Built-in Correlation）

所谓的内建关联规则通俗的说就是 LoadRunner 内部自带的一些规则。VuGen 针对常用的一些应用系统，如 AribaBuyer、BlueMartini、BroadVision、InterStage、mySAP、NetDynamics、Oracle、PeopleSoft、Siebel、SilverJRunner 等内建了很多关联规则，在 Tools→Recording Options→HTTP Properties→Correlation 中可以看到，如图 6-48 所示。在录制前可以启动需要的规则，这样在录制时，VuGen 会在脚本中自动建立关联。

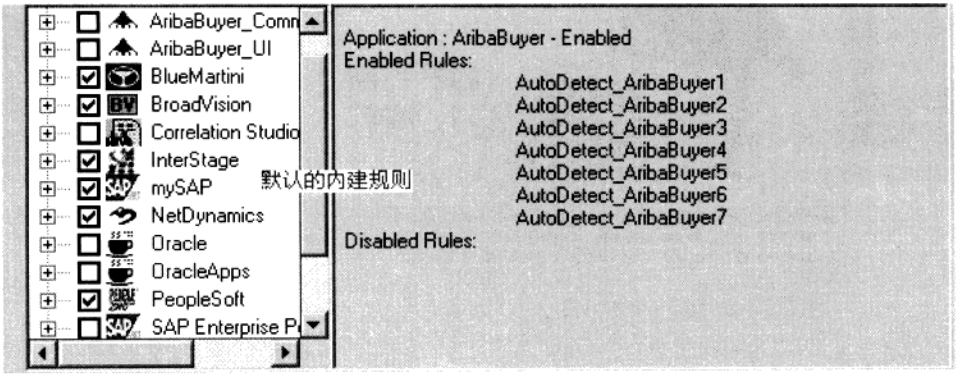


图 6-48 内建规则

（2）用户自定义关联规则（User-defined Rules Correlation）

如果在录制前发现默认的内建规则并不能满足录制需要，并且在录制前就已经知道规则的左右边界时，此时可以自己动手新建一个规则，如图 6-49 所示。点击 New Application 按钮先新建一个应用，再点击 New Rule 按钮为该应用新建一个规则，新建规则时只要正确填好左右边界信息即可。

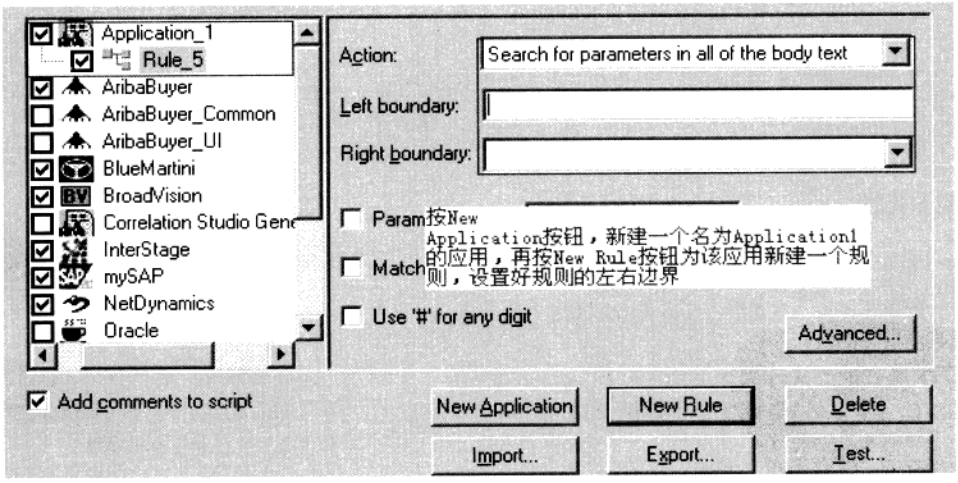


图 6-49 新建关联规则

2. 规则使用

在 Tools→Recording Options 对话框中选中 Enable correlation during recording 复选框，启用自动关联。在录制过程中，当 VuGen 检查到符合关联规则的数据时，会依照设定建立关联，产生与图 6-50 类似的一段脚本。

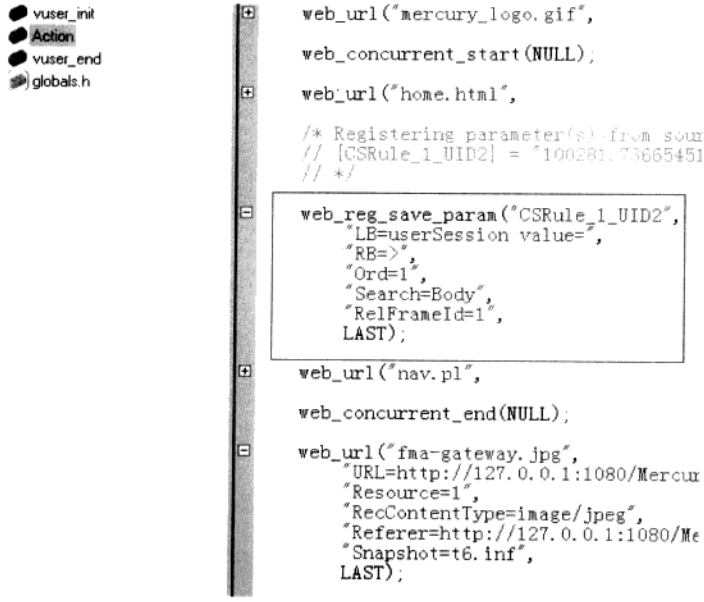


图 6-50 录制中关联

6.4.2 录制后关联

当录制前内建关联规则和新建规则都不能满足需要时，或者是不知道哪个地方需要关联时，只能采取录制后进行关联。

录制后关联与内建关联还是有点区别的，录制后关联是在执行脚本后才会建立关联，也就是说，当录制完脚本后，脚本至少要执行一次，录制后关联才会产生效果。录制后会尝试找到录制与执行时服务器响应的差异部分，找到需要关联的数据，并建立关联。

在脚本回放报错时，如图 6-51 所示，点击菜单 Vuser→Scan Script for Correlations 或按快捷键 Ctrl+F8，如图 6-52 所示。

```
Action.c(65): web_url("fma-gateway.jpg") was successful, 46063 body bytes, 167 header bytes
Action.c(73): web_url("mer_login.gif") was successful, 679 body bytes, 164 header bytes
Action.c(83): web_submit_data("login.pl") was successful, 748 body bytes, 225 header bytes
Action.c(99): web_concurrent_start was successful    [MsgId: MMSG-26392]
Action.c(101): Registering web_url("nav.pl_2") was successful    [MsgId: MMSG-26390]
Action.c(110): Registering web_reg_find was successful       [MsgId: MMSG-26390]
Action.c(113): Registering web_url("login.pl_2") was successful      [MsgId: MMSG-26390]
Action.c(122): Error -26366: "Text=arivn" not found for web_reg_find     [MsgId: MERR-26366]
Action.c(122): web_concurrent_end highest severity level was "ERROR", 2328 body bytes, 478 h
Ending action Action.
Ending iteration 1.
Ending Vuser...
Starting action vuser_end.
Ending action vuser_end.
```

图 6-51 未关联报错

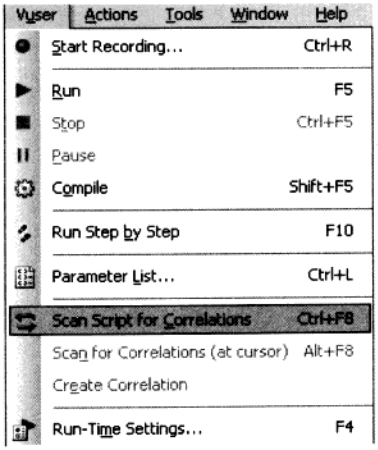


图 6-52 扫描脚本并关联

弹出关联信息结果，如图 6-53 所示。

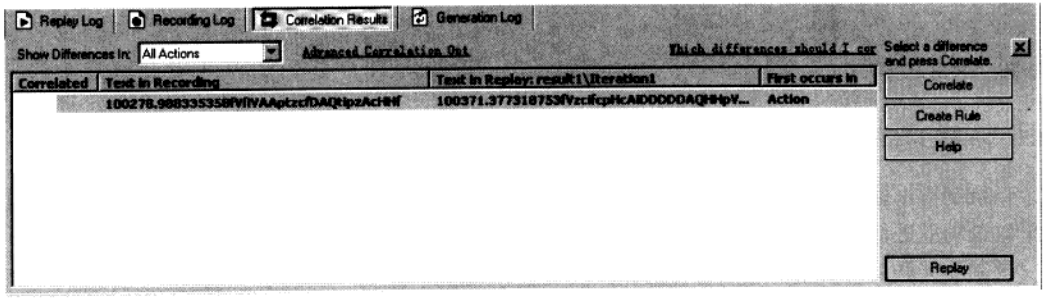


图 6-53 关联结果信息

检查扫描的结果，选择需要关联的数据，点击 Correlate 按钮，创建一个关联，这时被关联的数据前面会多出一个绿色的勾，如图 6-54 所示。

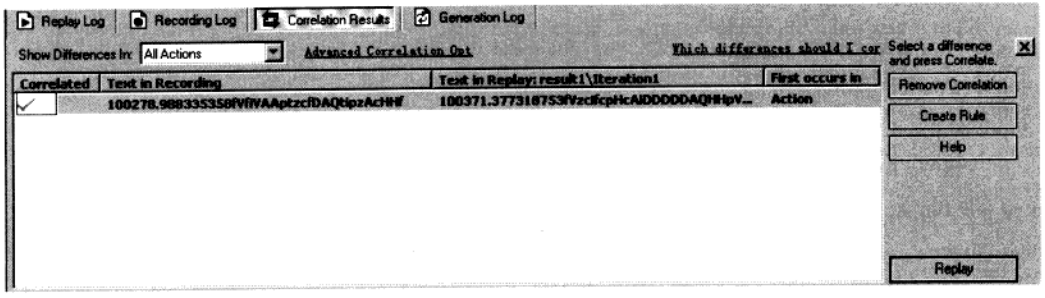


图 6-54 创建关联

再点击 Create Rule 按钮为关联创建规则，弹出 WebStudioViews 对话框，提示规则的左边界与右边界信息，点击“是”按钮即可，如图 6-55 所示。

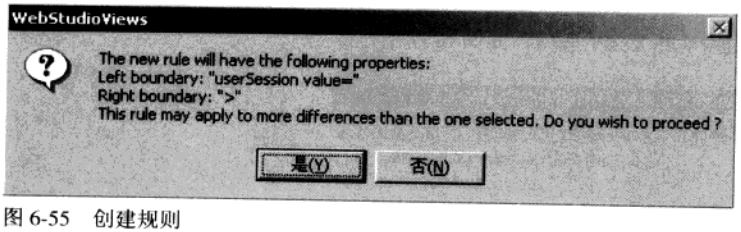


图 6-55 创建规则

这样关联就已经创建完成，在 Tools→Recording Options→HTTP Properties→Correlation 中可以看到刚才创建的规则，如图 6-56 所示。

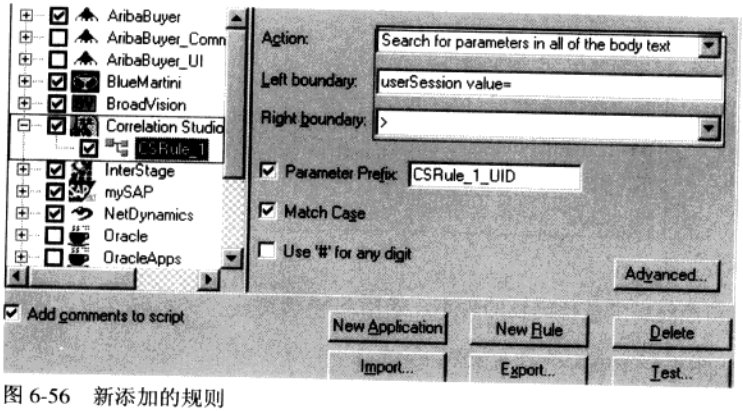


图 6-56 新添加的规则

图 6-56 中规则的左右边界与图 6-55 中弹出的提示信息左右边界信息一致。

6.4.3 手动关联

录制前关联与录制后关联都属于自动关联的范畴，正常情况下，自动关联就能解决脚本录制中大多数关联问题，但是如果出现使用上面两种方法都不能解决的情况，那么只能使用手动关联的方法进行关联了。手动关联和自动关联的原理是一样的，手动关联需要先找到需求关联的量，然后使用 LoadRunner 提供的关联函数进行关联。

手动关联的一般步骤如下：

1）录制两份脚本，保证业务流程和使用的数据相同。

简单的说，就是在录制的过程中使用相同的录制动作，录制两份脚本。

2）使用 WinDiff 工具比较两份脚本，找到需要关联的数据。

WinDiff 是 LoadRunner 自带的文件比较工具，用于比较两个文件的内容，找出两者之间不同

的地方，对两份脚本中不同的地方进行判断，进而找到需要关联的数据。这里为什么需要比较，是因为需要关联的数据都是从服务器端返回的数据，如果在比较中发现，对于同一个参数，服务器返回的值不一致，那么说明这个数据是动态的，需要进行关联。

在 Tool 菜单下选择 Compare with Vuser 选项，如图 6-57 所示。

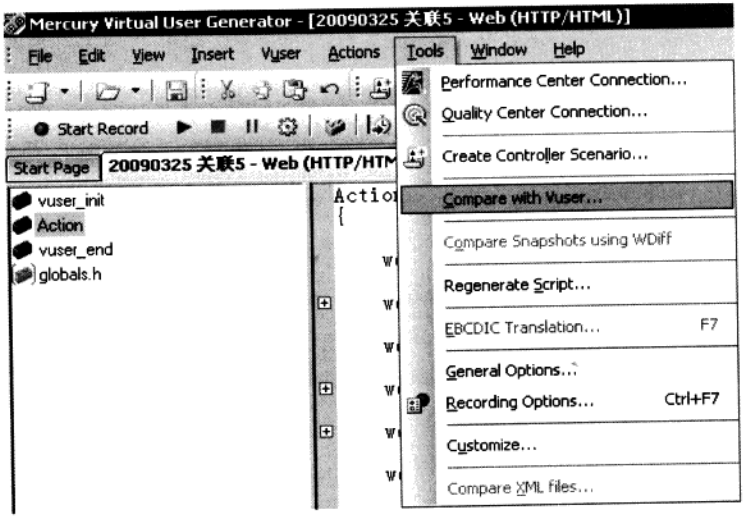


图 6-57 Compare with Vuser 选项

在弹出的对话框中选择要和当前脚本进行比较的脚本，如图 6-58 所示。

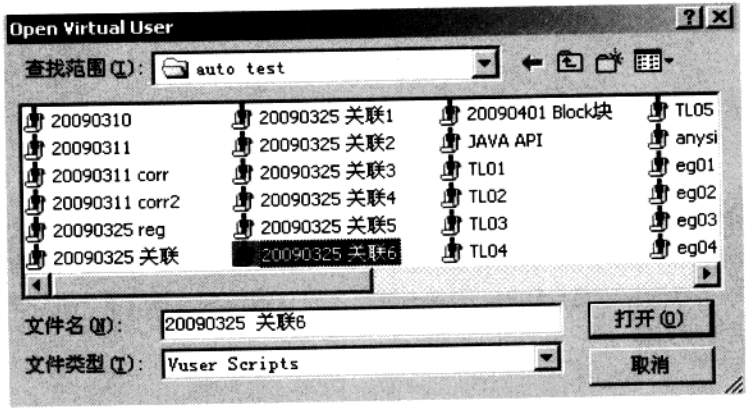
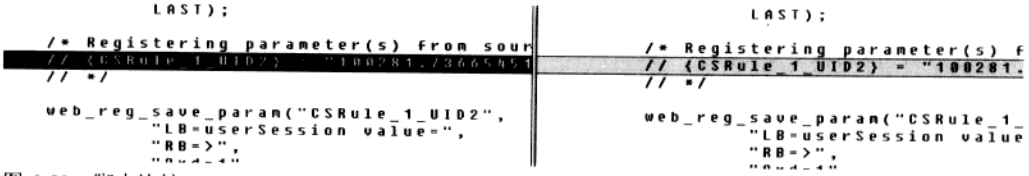


图 6-58 选择比较脚本

接着 WinDiff 启动，同时显示两份脚本，并显示有差异的地方，WinDiff 会以一整行黄色标示有差异的脚本，如图 6-59 所示。



```
        LAST);

    /* Registering parameter(s) from sour
    // {CSRule_1_UID2} = "100281./3665451
    // */

    web_reg_save_param("CSRule_1_UID2",
            "LB=userSession value=",
            "RB=>",
            "Ord=1"
```

```
        LAST);

    /* Registering parameter(s) f
    // {CSRule_1_UID2} = "100281.
    // */

    web_reg_save_param("CSRule_1_
            "LB=userSession value
            "RB=>",
            "Ord=1"
```

图 6-59 脚本比较

然后检查两份脚本中存在差异的地方，每个差异的地方都有可能是需要关联的地方。注意：lr_think_time 的差异部分不需要分析，因为 lr_think_time 是用来模拟每个步骤之间使用者思考延迟的时间。

3）找到左边界和右边界字符串。在插入关联函数之前一定要先找到左边界和右边界字符串，选择将 WinDiff 比较中不同的字符拷贝出来，在 Generation Log 中找到其对应的位置，如图 6-60 所示。

```
blockquote {font-family: tahoma; font-size : 10pt}
H1 {font-family: tahoma; font-size : 22pt; color: #993333}
H3 {font-family: tahoma; font-size : 10pt; color: black}
small {font-family: tahoma; font-size : 8pt}
</style>
<form method=post action=login.pl target=body>
<input type=hidden name=userSession value=100278.988385358fVfiVAAptzcfIAQtipzA:HHf>
<table border=0><tr><td> </td>
<td> </td>
<tr><td> </td>
<td> </td></tr>
<tr><td> </td>
<td><small> Username</small></td></tr>
<tr><td> </td>
<td><input type=text name=username value='' size=14 maxlength=14></td></tr>
<tr><td> </td>
<td><small> Password</small></td>
<tr><td> </td>
<td><input type=password name=password value='' size=14 maxlength=14></td></tr>
<tr><td> </td>
<td> <tr><td>
```

图 6-60 查找左边界与右边界

图中清楚地显示了关联数据的左边界与右边界信息，左边界为“userSession value=”，右边界为“>”，当然这里的左边界与右边界写得越长越好。这样关联函数就可以写出来了，如下。

```
web_reg_save_param("CSRule_1_UID2",
        "LB=userSession value=",
        "RB=>",
        "Ord=1",
        "RelFrameId=1",
        "Search=Body",
        LAST);
```

4）使用 web_reg_save_param 函数手动建立关联。规则的左边界与右边界信息找到后，需要插入 web_reg_save_param 函数来手动建立关联，首先找到关联函数插入的位置，点击 Vuser→Run-Time

Settings→General→Log→Extended log 将下面所有选项都选中。在 Replay Log 中找到 WinDiff 中比较不同的字符串，如图 6-61 所示，这段代码前面就是插入关联函数的位置。

```
web_url("nav.pl",

web_concurrent_end(NULL);

web_url("fma-gateway.jpg",          这行代码前面为要插入关联函数的位置

web_url("mer_login.gif",

lr_think_time(5);
```

Replay Log | Recording Log | Correlation Results | Generation Log

```
ction.c(63):     H3 {font-family: tahoma; font-size : 10pt; color: black}\n
ction.c(63):     small {font-family: tahoma; font-size : 8pt}\n
ction.c(63):     </style>\n
ction.c(63):     <form method=post action=login.pl target=body>\n
ction.c(63):     <input type=hidden name=userSession value=100372.964520049fVzcQDtpDzcfDAQHHpzAzVHf>\n
ction.c(63):     <table border=0><tr><td> </td>\n
ction.c(63):     <td> </td>\n
ction.c(63):     <tr><td> </td>\n
ction.c(63):     <td> </td></tr>\n
```

图 6-61 关联函数插入位置

5）将脚本中该关联数据以参数取代。当使用 web_reg_save_param 建立参数后，接下来就是用 CSRule_1_UID2 参数去取代脚

本中写死的（hard-coded）资料，最后参数的结果如图 6-62 所示。

```
web_reg_save_param("CSRule_1_UID2",
    "LB=userSession value=",
    "RB=>",
    "Ord=1",          关联函数
    "RelFrameId=1",
    "Search=Body",
    LAST);

web_url("nav.pl",

web_concurrent_end(NULL);

web_url("fma-gateway.jpg",

web_url("mer_login.gif",

lr_think_time(6);

web_submit_data("login.pl",
    "Action=http://127.0.0.1:1080/mercuryWebTours/login.pl",
    "Method=POST",
    "RecContentType=text/html",
    "Referer=http://127.0.0.1:1080/mercuryWebTours/nav.pl?in=home",
    "Snapshot=t9.inf",
    "Mode=HTTP",
    ITEMDATA,
    "Name=userSession", "Value={CSRule_1_UID2}", ENDITEM,
    "Name=username", "Value=arivm", ENDITEM,
    "Name=password", "Value=02321408", ENDITEM,
    "Name=JSFormSubmit", "Value=off", ENDITEM,
    "Name=login.x", "Value=0", ENDITEM,
    "Name=login.y", "Value=0", ENDITEM,
    LAST);
```

使用参数CSRule_1_UID2取代原来的常量

```
"Value=100278.988335358fVfiVAAptzcfDAQtipzAcHHf"
```

图 6-62 参数取代常量

7

场景设计实践

前面已经讲述了场景设计的一些基础知识，鉴于场景设计在整个性能测试过程中的重要性，在实际使用过程中也就存在很多使用技巧或者说是一些高级使用方法。通过这些技巧或高级使用方法可以帮助更好地进行场景设计或监控场景执行的情况。本章主要对场景设计过程中常用的一些技巧进行讲解，主要涉及的内容有：

- 集合点设置
- IP 欺骗技术
- 负载均衡技术
- 执行路径转换
- 在 LoadRunner 中使用功能测试脚本

7.1 集合点设置

首先需要明确在性能测试过程中为什么要进行集合点设置，在 LoadRunner 测试过程中，其实并不能保证所有的 Vuser 真的在同一时刻进行操作，这样就达不到并发测试的目的，故需要使用到集合点技术，集合点的意思是如果在一个操作之前设置了一个集合点，LoadRunner 会等待所有的 Vuser 都准备好要执行该功能时才开始执行，其强调的是所有的 Vuser 都已准备好了，如果只是部分 Vuser 准备好了，该功能还是不会被执行。

选择 Scenario→Rendezvous，如图 7-1 所示。

在场景设置集合点之前，脚本一定要插入集合点，否则会发现 Scenario 菜单中 Rendezvous 项是不可选的。

接下来会弹出集合点信息对话框，如图 7-2 所示。

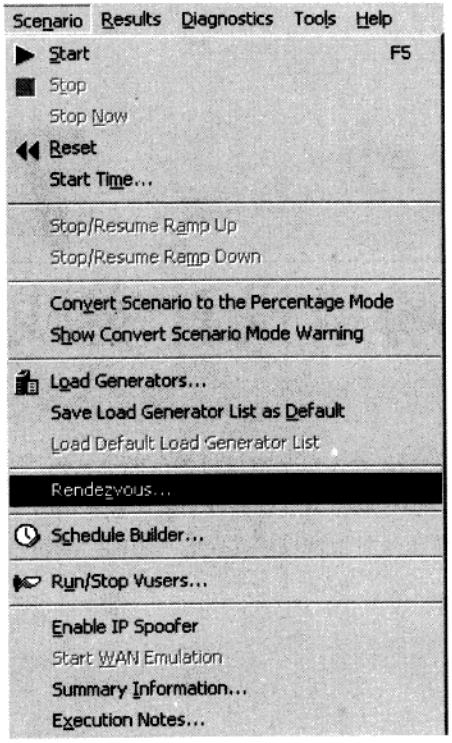


图 7-1 设置集合点操作

- Rendezvous：这里显示出脚本中包含的所有集合点。默认情况下这些集合点处于启用状态。可以点击 Disable Rendezvous 按钮，将集合点设置为禁用状态。
- Scripts：显示了场景运行的所有脚本。
- Vusers：场景运行设置虚拟用户情况。默认情况下所有的 Vuser 都会参与到集合点的策略中来，可以手动点击 Disable VUser 按钮，将一些 Vuser 设置为不参与到该集合点策略中来。

点击 Policy... 按钮，系统弹出 Policy 对话框，如图 7-3 所示。在该对话框中可以设定集合点执行的策略。

第一项：表示当所有用户数的 X%到达集合点时，开始释放等待的用户并继续执行场景。

第二项：表示当前正在运行用户数的 X%到达集合点时，开始释放等待的用户并继续执行场景。

第三项：表示当 X 个用户到达集合点时，开始释放等待的用户并继续执行场景。

其中还有一项 Timeout between Vusers，就 30 秒来说，当第一个用户到达集合点后，再等待 30 秒，如果在 30 秒内到达的用户数达到指定的数量，就开始继续执行场景。如果在 30 秒内还没有达到指定的用户数量，就不再等待，开始释放等待的用户并继续执行场景。

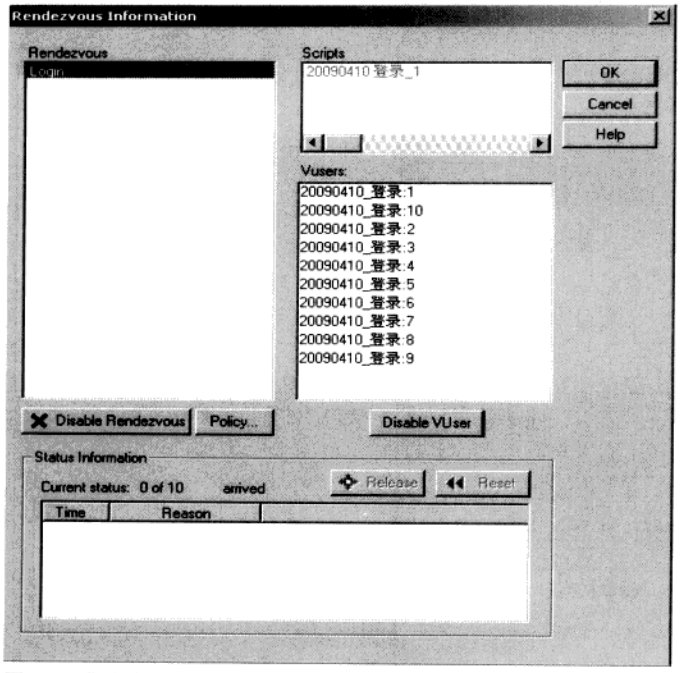


图 7-2 集合点设置

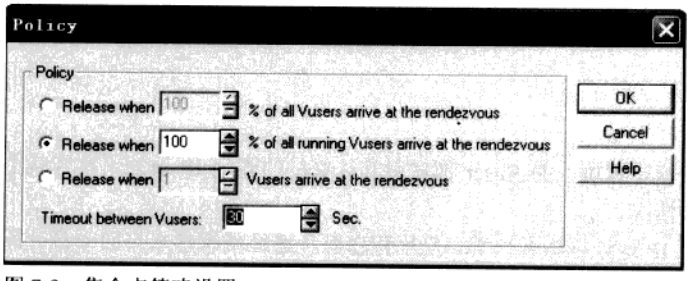


图 7-3 集合点策略设置

以上都是自动控制 Vuser 释放的情况，但在实际使用的过程中，也可以手动对 Vuser 进行释放，这涉及到手动释放 Vuser 技术。在正常情况下会发现在集合点信息对话框中，下面的 Release 按钮不可用，如图 7-2 所示。以下是手动释放 Vuser 的步骤：

1）开始执行场景。

2）在场景执行过程中，选择 Scenario→Rendezvous，打开集合点信息对话框。

3）在场景运行过程中，下面的 Release 按钮和 Reset 按钮会变成可用状态，这时可以点击 Release 按钮，进行手动释放 Vuser，如图 7-4 所示。

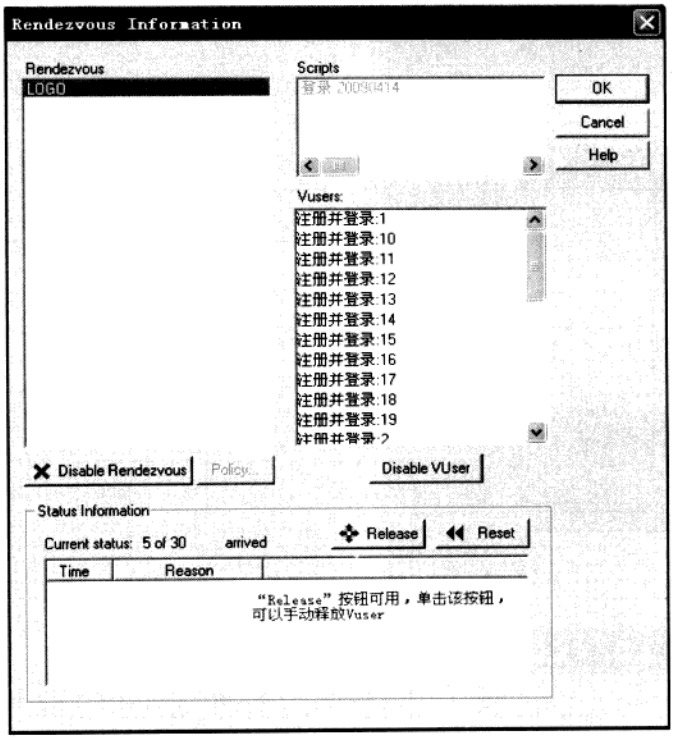


图 7-4 手动释放 Vuser

7.2 IP 欺骗技术

在场景运行时，每台负载发生器计算机上的 Vuser 都使用其计算机的固定 IP 地址。这样就不能模拟用户使用不同计算机的真实情况。

应用程序服务器和网络设备用 IP 地址标识客户端。应用程序服务器经常缓存来自同一台计算机的客户端信息。网络路由器则尝试缓存源信息和目标信息来优化吞吐量。如果许多用户使用同一个 IP 地址，服务器和路由器都将尝试进行优化处理。由于同一台负载发生器计算机上的 Vuser 具有相同的 IP 地址，因此服务器和路由器优化不反映真实情况。

使用 LoadRunner 的多 IP 地址功能，可以用许多 IP 地址来标识在一台计算机上运行的多个 Vuser。这样，服务器和路由器将认为 Vuser 来自不同的计算机，因此测试环境更加真实。

当然 LoadRunner 在设计的时候就想到这个问题了，这就是 IP 欺骗技术。但是在使用 IP 欺骗时需要注意 IP Spoofer 要在连接 Load Generator 之前启用，并且各负载发生器计算机必须使用固定 IP，不能使用动态 IP（即 DHCP）。

7.2.1 IP Spoofer 设置

1）点击“开始”菜单→“所有程序”→Mercury LoadRunner→Tools→IP Wizard，弹出 IP Wizard 配置对话框，如图 7-5 所示。

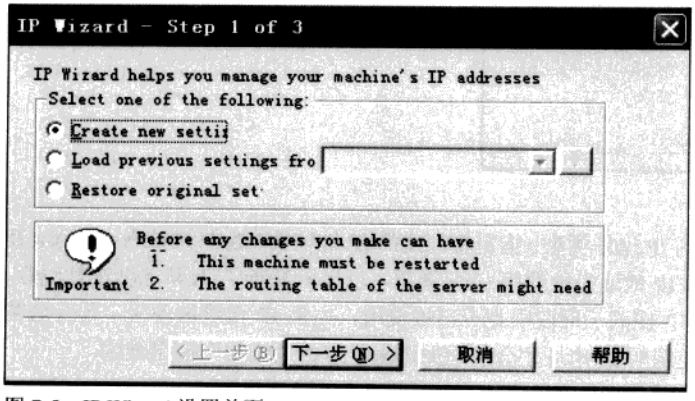


图 7-5 IP Wizard 设置首页

2）在这里选择一种方式进行设置，如果以前保存过这类文件，可以选择“Load Previous settings from（从文件中加载以前的设置）”，然后选择该文件即可。

3）也可以选择“Create new settings（创建一个新的设置）”。

4）点击“下一步”按钮。如果计算机中安装了多个网卡，那么要选择用于这些 IP 地址的网卡，然后点击“下一步”按钮，在弹出的对话框中设置服务器的 IP 地址，如图 7-6 所示。

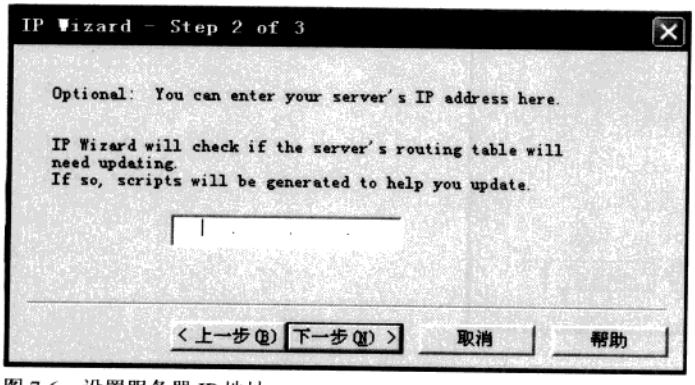


图 7-6 设置服务器 IP 地址

5）点击“下一步”按钮将看到该计算机的 IP 地址列表。点击“添加”按钮可以定义地址范围，如图 7-7 所示。

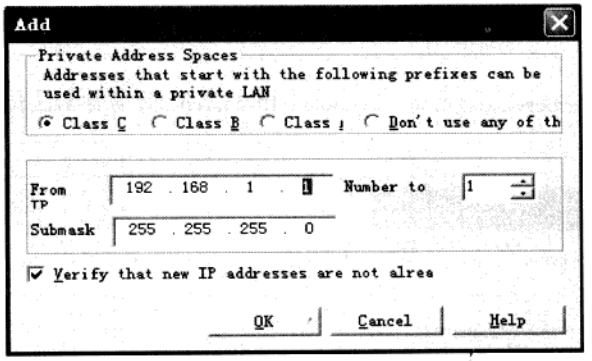


图 7-7 设置虚拟 IP 和子网掩码

6）在该对话框中选择计算机的 IP 地址类型。指定要创建的 IP 地址数。选中“Verify that new IP addresses are not already（验证新的 IP 地址未被使用）”复选框，以指示 IP 向导对新地址进行检查，LoadRunner 会自动检查新添加的 IP 在同一网段中是否已被使用，如果 IP 已经被使用，那么这些 IP 将不会被添加进来，只有未被使用的 IP 地址才会被添加进来，点击“确定”按钮继续。

7）完成之后，IP 欺骗向导会显示出 IP 变更统计的对话框，如图 7-8 所示。

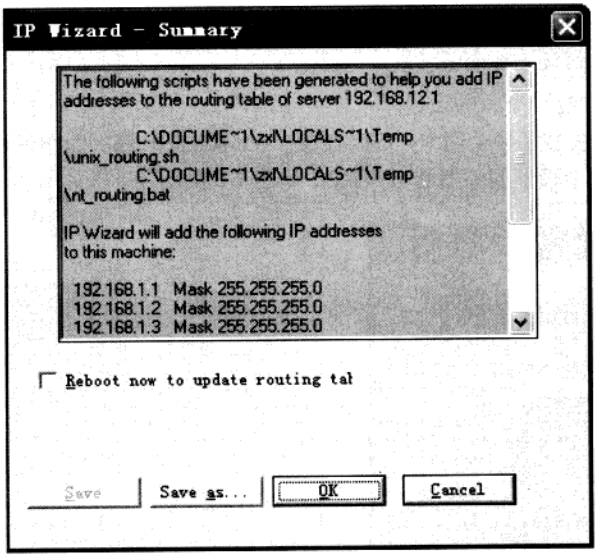


图 7-8 IP 变更统计

在该对话框中，可以点击 Save as... 按钮将本次添加的 IP 保存成“.ips”文件，下次再使用时就可以直接加载此文件。

完成之后，系统会提示重启计算机，重启计算机后，设置的 IP 欺骗有效，重启后为了确认虚

拟 IP 是否都生效，可以在“开始”→“运行”，输入“cmd”，在弹出的命令窗口输入“ipconfig/all”，就能看到生效的所有 IP。

还可以使用 lr_get_vuser_ip 函数来得到当前虚拟用户的 IP 地址，在脚本中加入下面的代码即可。

```
char *ip;
ip = lr_get_vuser_ip();
if(ip)
lr_output_message("The IP address is %s",ip);
else
lr_output_message("IP spoofing disabled");
```

场景运行时，如果虚拟 IP 生效的话，打开每个 Vuser 的日志，就可以看见他们各自的 IP。

而在实际使用过程中，经常发现，客户端和服务器之间存在一个路由器，出现这种情况怎么办呢？其实在使用 IP 欺骗技术时，服务器需要将这些地址添加到路由表中，以识别返回客户端的路由。如果服务器和客户端具有相同的子网掩码、IP 类和网络，那么不需要修改服务器的路由表。如果客户端和服务器之间有路由器，那么必须手动让服务器将这些地址添加到路由表中，这样服务器才能识别经过该路由器的路径。

在设置完成 IP 欺骗的最后一步，不要重启计算机，此时在临时文件夹 Temp 里生成两个文件，一个是用于 UNIX 操作系统的 unix_routing.sh 文件，一个是用于 Windows 操作系统的 nt_routing.bat 文件。两个文件的内容如下：

nt_routing.bat 文件：

```
REM This is a bat file to add IP addresses to the routing table of a server
REM Replace [CLIENT_IP] with the IP of this machine (Load Generator) that the server already recognizes
REM This script should be executed on the server machine
route ADD 192.168.14.22 MASK 255.255.255.255 [CLIENT_IP] METRIC 1
route ADD 192.168.14.23 MASK 255.255.255.255 [CLIENT_IP] METRIC 1
route ADD 192.168.14.25 MASK 255.255.255.255 [CLIENT_IP] METRIC 1
```

unix_routing.sh 文件：

```
# Bourne shell script to add IP addresses to the routing table of a server
# To run replace [CLIENT_IP] with the IP of this machine (Load Generator) that the server already recognizes
# Then chmod +x unix_routing.sh
# and finally execute this script on the server
#!/bin/sh
route add 192.168.14.22 [CLIENT_IP] 255.255.255.0
route add 192.168.14.23 [CLIENT_IP] 255.255.255.0
route add 192.168.14.25 [CLIENT_IP] 255.255.255.0
```

要更新服务器路由表，必须对这两个文件进行一定的修改。

1）将[CLIENT_IP]用 LoadRunner 机器的 IP 地址替换。假设 LoadRunner 机器的 IP 地址为 192.168.14.26。

2）修改的文件必须在服务器上运行该文件，Windows 操作系统使用 nt_routing.bat 文件，UNIX 操作系统使用 unix_routing.sh 文件。

修改后的两个文件内容如下：

nt_routing.bat 文件：

```
REM This is a bat file to add IP addresses to the routing table of a server
REM Replace [CLIENT_IP] with the IP of this machine (Load Generator) that the server already recognizes
REM This script should be executed on the server machine
route ADD 192.168.14.22 MASK 255.255.255.255 192.168.14.25 METRIC 1
route ADD 192.168.14.23 MASK 255.255.255.255 192.168.14.25 METRIC 1
route ADD 192.168.14.25 MASK 255.255.255.255 192.168.14.25 METRIC 1
```

unix_routing.sh 文件：

```
# Bourne shell script to add IP addresses to the routing table of a server
# To run replace [CLIENT_IP] with the IP of this machine (Load Generator) that the server already recognizes
# Then chmod +x unix_routing.sh
# and finally execute this script on the server
#!/bin/sh
route add 192.168.14.22 192.168.14.25 255.255.255.0
route add 192.168.14.23 192.168.14.25 255.255.255.0
route add 192.168.14.25 192.168.14.25 255.255.255.0
```

到这里跨网段 IP 欺骗技术已经完成，重启计算机，IP 欺骗将生效。

7.2.2 Controller 中启动 IP Spoofer

设置好 IP Spoofer 后，必须在 Controller 中启动 IP Spoofer，选择 Scenario→Enable IP Spoofer，启动 IP Spoofer 策略，如图 7-9 所示。

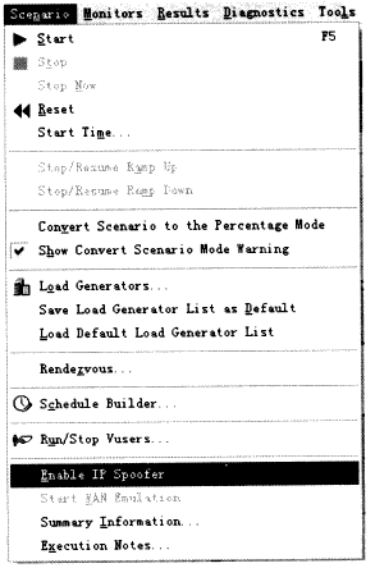


图 7-9 启动 IP 欺骗策略

当启动 IP Spoofer 后，在 Run 选项卡的右下角会看到 IP Spoofer 的标记，如图 7-10 所示。

图 7-10 IP Spoofer 启动标记

启动多 IP 地址模式后，可以在专家模式中对多 IP 地址进行全局设置。首先，选择 Tools→Expert Mode 来启动专家模式。接下来选择 Tools→Options→General 选项卡，来设置多 IP 地址模式分配 IP 地址的情况，如图 7-11 所示。

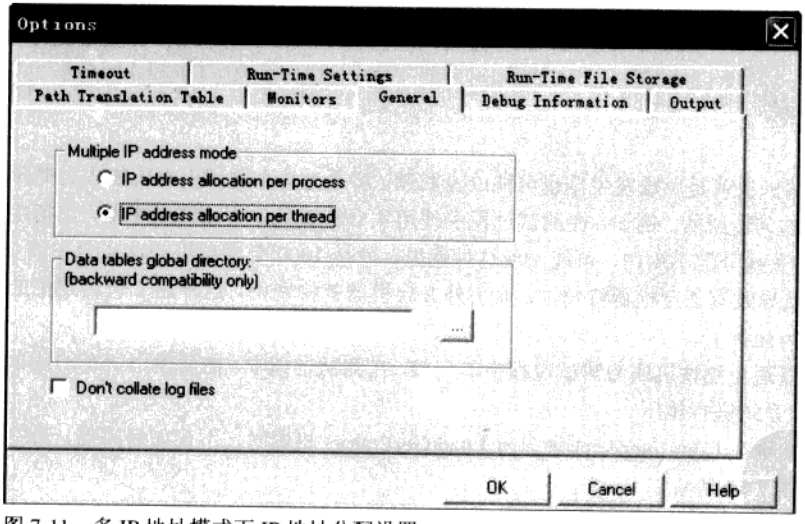


图 7-11 多 IP 地址模式下 IP 地址分配设置

可以选择按进程或线程分配 IP 地址，按线程分配会导致场景中存在更多的 IP 地址范围。

选择 Scenario→Load Generators，设置 Vuser 生成器，将虚拟 IP 地址添加进去，并连通（点击 Connect 按钮，每个 IP 状态由 Down 变为 Ready），如图 7-12 所示。

当连接成功后，Vuser 生成器会在工具栏中显示，如图 7-13 所示。

在使用完成之后，一定要记得释放 IP 地址，在 IP Wizard 设置首页，如图 7-5 所示。选择 Restore original set 项，进行 IP 地址释放，重启计算机即可。

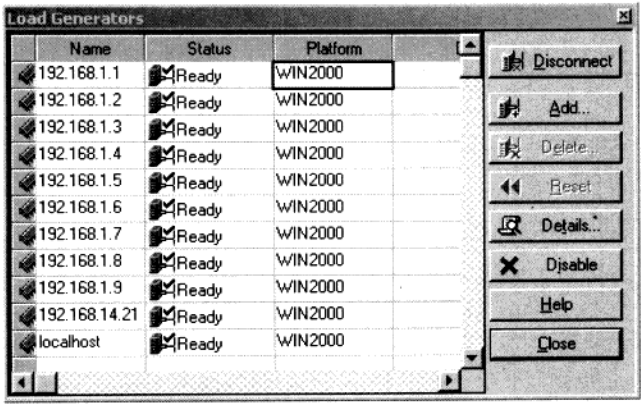


图 7-12 Vuser 生成器设置

图 7-13 连接成功标识

7.3 负载均衡技术

在测试过程中，测试机也就是负载发生器也可能成为瓶颈。那么为什么测试机也可能成为瓶颈呢？原因是由于负载不均匀造成的。例如，在测试过程中使用 4 台测试机作为负载发生器，虚拟用户为 1000 个，这时如果负载分配不均匀，可能出现这种现象，就是 1000 个虚拟用户从 4 台中的 2 台测试机中产生，这样就导致有 2 台机器特别忙，而另外 2 台机器就特别闲，这样这 2 台特别忙的机器，其本身就可能成为瓶颈了。

为了尽可能减少或者避免测试机成为测试过程中的瓶颈，在测试过程中，需要使用所有的测试机产生 Vuser，对被测试系统进行施压。

在默认模式下，会发现在 Controller 中添加多台 Load Generators 机器时，不管如何添加，最终只能选中一台机器，如图 7-14 所示。

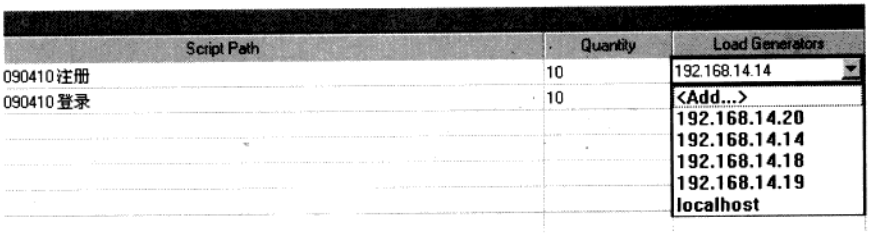


图 7-14 Load Generators 机器设置

这样的负载分配是不均匀的，为了解决这个问题，首先要更换场景模式，选择 Scenario→Convert Scenario to the Percentage Mode，将场景模式由组模式更换为百分比模式，如图 7-15 所示。在弹出的对话框中点击“是”按钮即可。

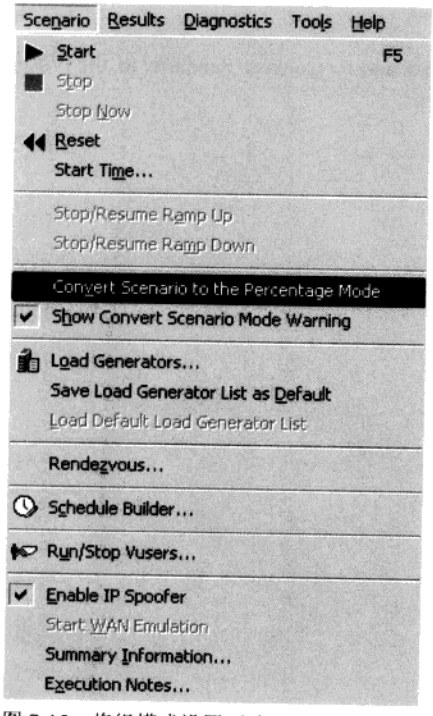


图 7-15 将组模式设置更改为百分比模式

这时，可以在已经添加好的 Load Generators 机器列表中选择需要的机器，如图 7-16 所示。

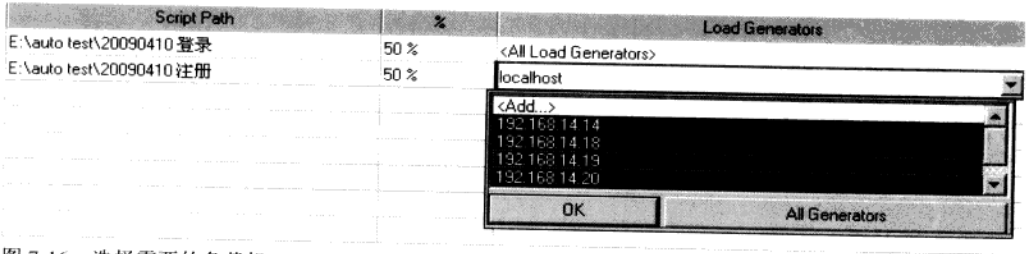


图 7-16 选择需要的负载机

选择需要的负载机，也可以选择所有的负载机，点击 OK 按钮，所选择的机器都将被添加进来，如图 7-17 所示。

Script Path	%	Load Generators
E:\auto test\20090410登录	50 %	<All Load Generators>
E:\auto test\20090410注册	50 %	192.168.14.20, 192.168.14.14, 192.168.14.18, 192.168.14.19, localho

图 7-17 为 Load Generators 设置多台机器

如果需要将百分比模式切换回组模式，那么可以选择 Scenario→Convert Scenario to the Vuser Group Mode，将场景模式切换为组模式，如图 7-18 所示。

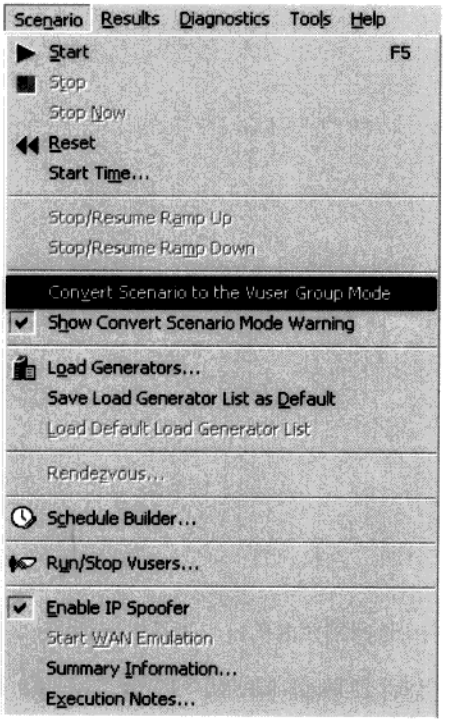


图 7-18 将百分比模式设置为组模式

这样就可以保证负载机均匀地对服务器进行施压。

7.4 执行路径转换

7.4.1 路径转换介绍

在运行场景时，LoadRunner 会从正在参与的 Vuser 中收集运行时的数据。默认情况下，

LoadRunner 将数据存储在每台 Vuser 计算机上的临时文件中。当场景运行完成后，会在常规结果目录中整理数据。但也可以设置为直接将运行时数据写入共享网络驱动器中。建议不要使用此方法，因为这样会增加网络流量且需要进行路径转换设置。

路径转换是 LoadRunner 在转换 Controller 的远程路径名时所使用的一种机制。一个远程负载发生器将网络驱动器映射为 F，而另一个负载发生器将同一个驱动器映射为 H。在如此复杂的场景中，需要确保所有参与的计算机能够识别同一个网络驱动器。

在 Tools→Options→Run-Time File Storage 选项卡中，可以设置将脚本和运行时数据结果存储在共享网络驱动器上，如图 7-19 所示。

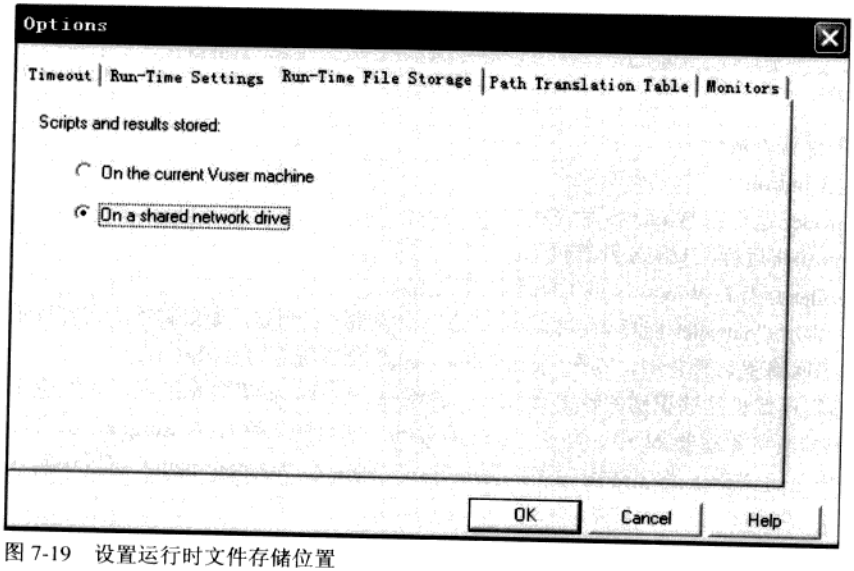


图 7-19 设置运行时文件存储位置

如果 Controller 和远程负载发生器对存储运行时结果文件的共享驱动器采用不同的映射，必须执行路径转换。在 Windows 和 UNIX 之间进行跨平台路径转换也是有效的。可以通过路径转换将基于 Windows 的路径（可以被 Controller 识别）转换成能被 UNIX Vuser 负载发生器识别的路径。

7.4.2 编辑路径转换表

LoadRunner 将路径转换表保存在一个 ASCII 文件（ppath.mnt）中。该文件存储在 LoadRunner 根目录/dat 中，对于每个要转换的网络路径都包含一行条目，如图 7-20 所示。

路径转换表中每一行条目的格式如下：

<controller_host> <controller_path> <remote_path> [<remote_host>]

controller_host：表示正在运行 Controller 的计算机名称或类型。例如，如果 Controller 运行在

一台 UNIX 计算机上，可以在“controller_host”字段中键入 unix。也可以输入 Controller 计算机的名称（如 PC1）。

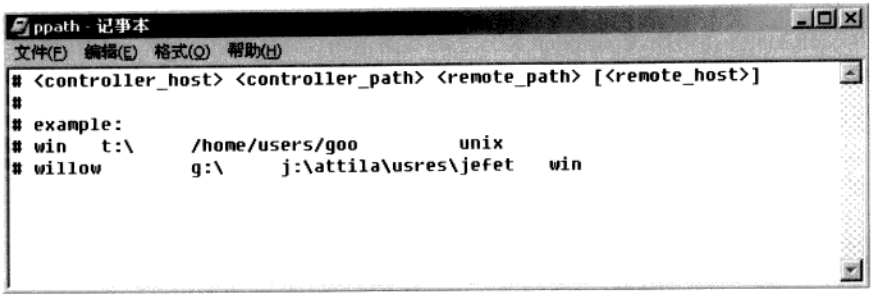


图 7-20 ppath 文件内容

controller_host 的值可以是：

- Hostname：Controller 运行的计算机名。
- Win：Controller 运行在 Windows 计算机上。
- Unix：Controller 运行在 UNIX 计算机上。
- All：Controller 运行在 Windows 或 UNIX 计算机上。

controller_path：表示 Controller 能够识别的特定目录所在的路径。例如，如果脚本所在的目录位于 Controller 映射的网络驱动器 R 中，则在“controller_path”字段键入路径 R:\scripts。

remote_path：远程计算机能够识别的特定目录所在的路径。例如，如果脚本所在的目录位于远程负载发生器映射的网络驱动器 N 中，则在“remote-path”字段键入路径 N:\scripts。如果远程 UNIX 负载发生器上的 Vuser 将上述路径识别为/m/tests，则应在“remote-path”字段中键入该路径。

remote_host：表示负载发生器的名称或类型。其他可设置的值和 controller_host 一致，该参数是可选参数。

下面看两个例子：

arivn F:\ G:\loadtest\ huang

表示 Vuser 运行在 Windows 计算机 huang 上。Arivn 将网络驱动器映射为 F:，而 huang 将其映射为 G:\loadtest。

arivn F:\ /u/tests/load Unitr

表示 Vuser 运行在 UNIX 计算机 Unitr 上。Arivn 将网络驱动器映射为 F:，而 Unitr 将其映射为 /u/tests/load。

接下来就是编辑路径转换表了，具体步骤如下：

1）在 Controller 控制器中，选择 Tools→Options，选择 Path Translation Table 选项卡，打开路径转换表视图，如图 7-21 所示。

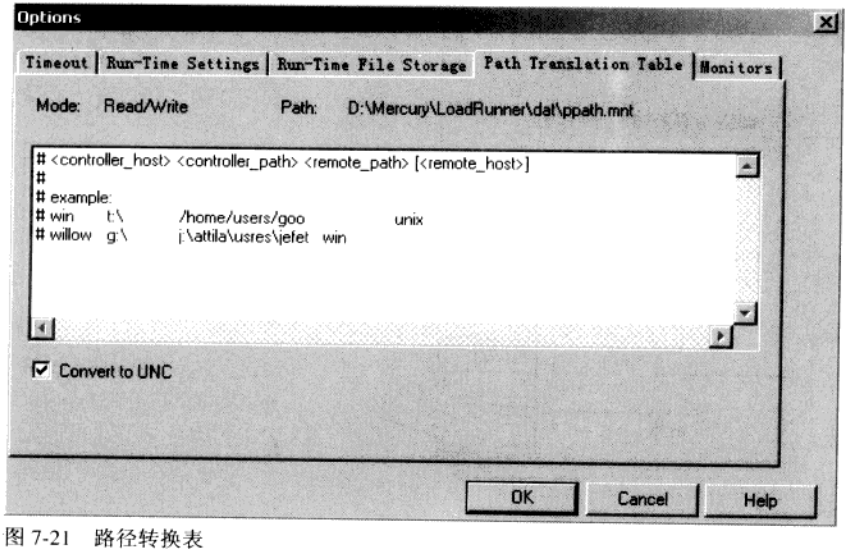


图 7-21 路径转换表

2）输入路径转换信息之前，请首先考虑使用通用命名约定方法。如果您的计算机是 Windows 计算机，可以指示 Controller 将所有路径转换为 UNC。这样，所有计算机都可以识别路径而无需进行路径转换。以下为 UNC 格式路径的一个示例：\\machine_a\results。选中 Convert to UNC 复选框，表示 LoadRunner 将忽略路径转换表并将所有路径都转换为通用命名约定格式。

3）如果您的计算机不是 Windows 计算机并需要路径转换，则在该表中键入路径信息。可以通过在该表中行首处键入#符号来插入注释。点击 OK 按钮保存信息即可。

7.5 在 LoadRunner 中使用功能测试脚本

LoadRunner 能够将 GUI Vuser 脚本形式的功能测试脚本集成到负载测试场景中。这样就可以使用 LoadRunner 来测试和监控负载对应用程序功能的影响。LoadRunner 可以将 GUI Vuser 脚本形式的这些功能测试脚本集成到负载测试场景中，在 QuickTest 或 WinRunner 中设计并调试过的这些脚本可用作负载测试的基础。

在 LoadRunner 中运行功能测试脚本主要有以下几个优点：

- 可检查高负载对应用程序的功能带来的影响。
- 可以度量应用程序在负载下运行时，典型用户在客户端等待的响应时间（端到端的响应时间）。

通过 GUI Vuser 可以度量并监控客户端/服务器系统在负载下端到端的用户响应时间。端到端的响应时间表示用户在提交请求后等待响应的总时间。端到端的响应时间包括 GUI 响应时间、网络和服务器响应时间，如图 7-22 所示。

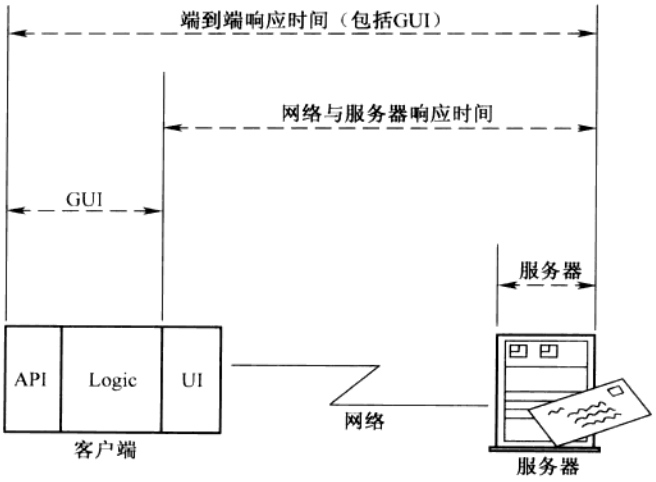


图 7-22 GUI 响应时间

7.5.1 QuickTest 创建 GUI Vuser 脚本

在 QuickTest 中创建用于 LoadRunner 测试场景中的 GUI Vuser 脚本时，需要遵循特定的准则以确保能够平衡集成脚本。QuickTest 具有多项与 LoadRunner 集成设计的功能。但部分 QuickTest 功能可能在与 LoadRunner 集成时不可用。QuickTest 与 LoadRunner 集成要注意以下几个方面。

1. 事务

在一些时候为了知道服务器对某一个业务的处理能力，需要定义事务。在 Vuser 脚本中可以通过嵌入适当的开始和结束事务语句脚本段来定义事务。但要注意的是 LoadRunner 仅提供事务中数据性能信息。因此，QuickTest 测试必须包括 LoadRunner 要使用的事务。通常在 QuickTest 中也使用插入开始事务和插入结束事务来插入事务。这样可以确保事务被正确地集成到 LoadRunner 中。

2. 将集成数据与测试一起保存

QuickTest 为了能与 LoadRunner 虚拟用户技术集成，必须生成特殊的集成文件。在默认情况下，生成这些数据的选项都是处于开启状态，但有时为了保留磁盘空间而禁用该选项。这样在录制时就要确保该选项处于开启状态。

在 QuickTest 中，选择 Tools→Options 对话框，打开 Run 选项卡，如图 7-23 所示。在这里一定要保证 Allow other Mercury products to run tests and components 选项处于选中的状态，如果没有选中，则将其选中并保存设置。

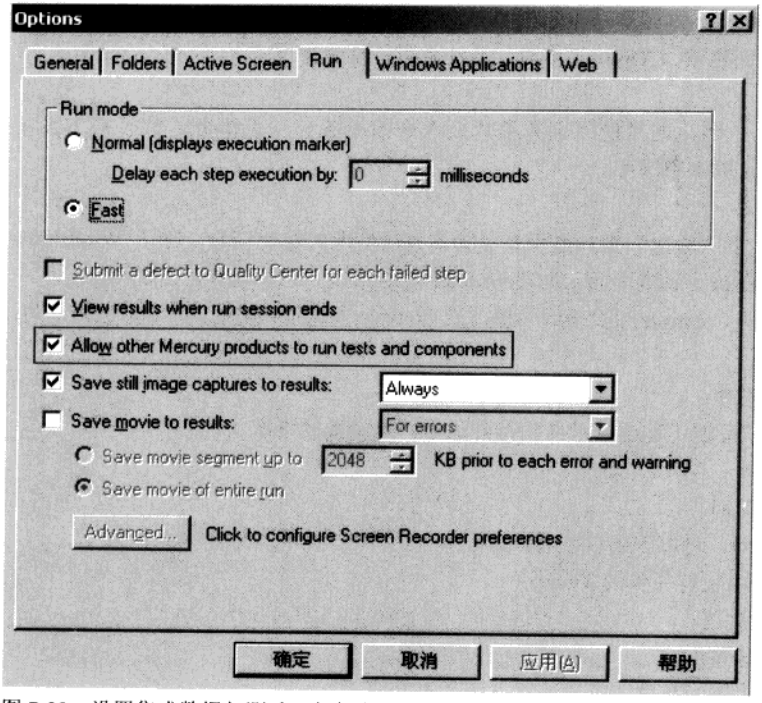


图 7-23 设置集成数据与测试一起保存

3. 添加语句

可以使用 Services 对象及其相关方法来插入与性能测试相关的语句，主要包括：Abort、GetEnvironmentAttribute、LogMessage、SetTransactionStatus、ThinkTime、UserDataPoint、StartTransaction 及 EndTransaction。

4. 为 LoadRunner 设计测试

在设计要用于 LoadRunner 的测试时，需要考虑以下设计准则：

- LoadRunner 中使用的 QuickTest 测试应该简单，并专门针对特定的操作。
- LoadRunner 无法运行嵌入式操作迭代。
- 请勿引用外部操作或其他外部资源，如外部数据表文件、环境变量文件和共享对象库等。

7.5.2 WinRunner 创建 GUI Vuser 脚本

WinRunner 是一个基于 Windows 平台，用于创建、编辑和调试 GUI Vuser 脚本的完整开发环境。通过 WinRunner 可记录真实用户在应用程序中的操作。

WinRunner 的 GUI Vuser 脚本是使用 TSL（Mercury Interactive 的测试脚本语言）编写的。TSL 是一种与 C 类似的高级编程语言。它结合了传统编程语言的功能和灵活性以及专为测试设计的功

能。但是 WinRunner 中创建的 Vuser 脚本是不能在 UNIX 计算机上运行的。

同样，使用 WinRunner 创建 GUI Vuser 脚本时，要注意以下几个方面：

1. 插入事务和集合点

插入事务和 QuickTest 一样，直接使用事务函数插入开始事务与结束事务，集合点方面将 Rendezvous 语句插入到 Vuser 脚本中即可。

2. 向 Controller 发送消息

运行场景时，Controller 的“输出”窗口显示有关脚本执行情况的重要信息。除了 WinRunner 自动发送的消息外，还可以在每个脚本中插入向 Controller 发送错误消息和通知消息的语句。

error_message 函数用于向 Controller 的“输出”窗口或 Console 发送错误消息。此函数的语法为：

error_message(message);

其中 message 为文本字符串。

output_message 函数用于发送不是错误消息的特殊通知。此函数的语法为：

output_message(message);

其中 message 为文本字符串。

在执行场景或会话步骤间，可以识别以下对象：

- 某特定时刻在场景中执行任务的 Vuser。
- 执行脚本的负载发生器。
- 运行 Controller 的计算机。

而有关 Vuser 和负载发生器的信息可以通过以下函数来获取：

lr_whoami：返回 Vuser 名称及其所属的 Vuser 组。

get_host_name：返回执行脚本的计算机名称。

get_master_host_name：返回运行 Controller 或 Console 的计算机名称。

7.5.3 场景中使用 GUI Vuser 脚本

在 QuickTest 或 WinRunner 中创建 GUI Vuser 脚本后，便可以将该脚本集成到 LoadRunner 场景中。在将 GUI Vuser 脚本添加到 LoadRunner 场景前，要注意以下几个问题：

- 每台计算机上每次只能运行一个 GUI Vuser。
- 请确保在运行场景前关闭 Q uickTest 或 WinRunner。
- 在 Run-time Settings for script 设置对话框中，只有 General 类中的 Run Logic 和 Think Time 与 QuickTest 和 WinRunner 测试有关，如图 7-24 所示。

接下来就是要将 GUI Vuser 添加到 LoadRunner 场景中，在控制器界面，选择 File→Open，在弹出的对话框中选择要添加的 GUI Vuser 脚本。也可以选择 File→New，在弹出的 New Scenario 对话框中点击 Browse... 按钮，弹出 Open Test 对话框，选择要添加的 GUI Vuser 脚本。这里有一个“文件类型”框，如图 7-25 所示。对于 WinRunner 来说，选择“GUI Scripts”文件类型。对于 QuickTest 来说，选择“Astra Tests”文件类型。确定文件类型后，选择合适的脚本并进行添加即可。

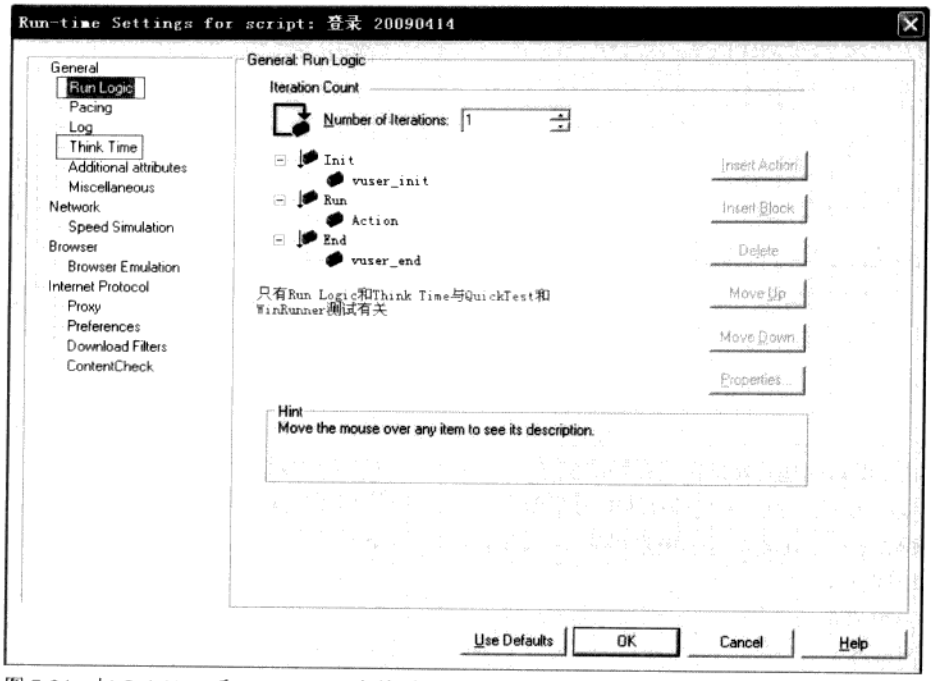


图 7-24 与 QuickTest 和 WinRunner 有关项

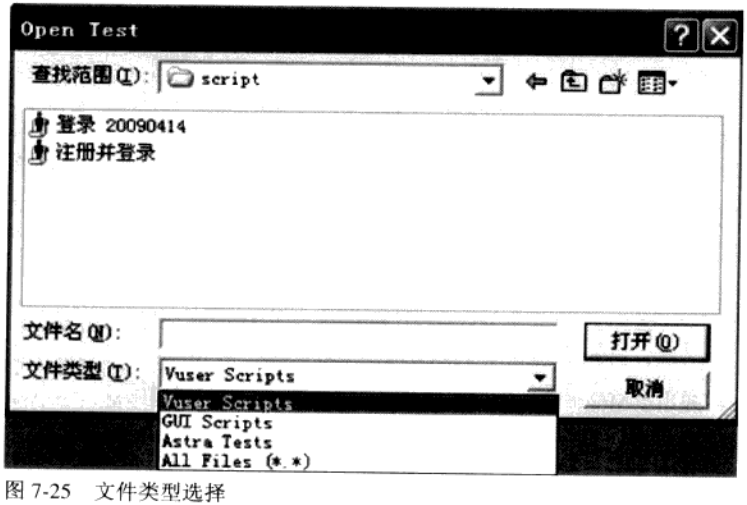


图 7-25 文件类型选择

8 结果分析实践

第 5 章对分析器的基础知识做了详细的讲述，但在真正的项目实践过程中仅仅依靠那些知识是远远不够的，必须借助一些更先进的技术来分析数据视图。只有这样才能更好地捕捉分析图的信息，更好地处理测试结果。在测试过程中常用的分析图的技术有以下几种：

- 分析图合并
- 分析图关联
- 页面细分
- 钻取技术
- 导入外部数据

8.1 分析图合并

分析器保存的图都是单个的图，在分析结果的过程中往往发现，仅仅靠单个图的分析是不够的。单个图只是从单个的度量角度去分析结果，并没有从多个角度去度量测试结果。这时就希望可以将有关系的一些图合并起来查看。使用 Analysis 可以将同一个方案中的两个数据图结果合并到一个数据图中。通过对分析图进行合并，可以同时从多个角度去度量结果并且可以观察这两个视图之间的关系。

8.1.1 分析图合并原理

选择 View→Merge Graphs，弹出“Merge Graphs（合并图）”对话框，如图 8-1 所示。

这里有 3 个需要设置的属性。

1）选择要合并的图：选择一个要与当前活动图合并的图。注意这里只能选择 X 轴度量单位相同的图。

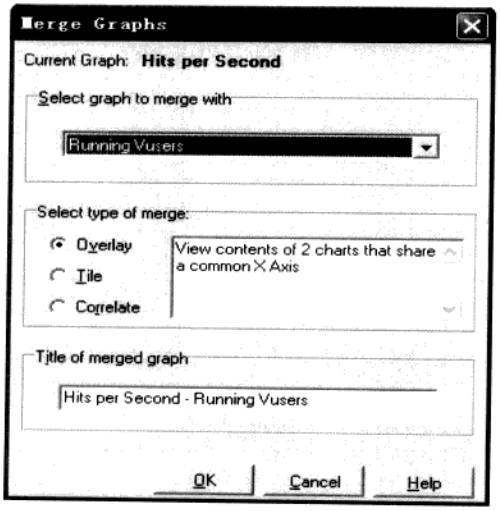


图 8-1 设置合并图

2）选择合并类型：有三种类型供选择：叠加、平铺和关联。这三种合并方式还是存在一些不同之处，具体如下：

- 叠加：查看共用同一 X 轴的两个图的内容。合并图左侧的 Y 轴显示当前图的 Y 轴值，右边的 Y 轴显示合并进来的图的 Y 轴值，如图 8-2 所示。

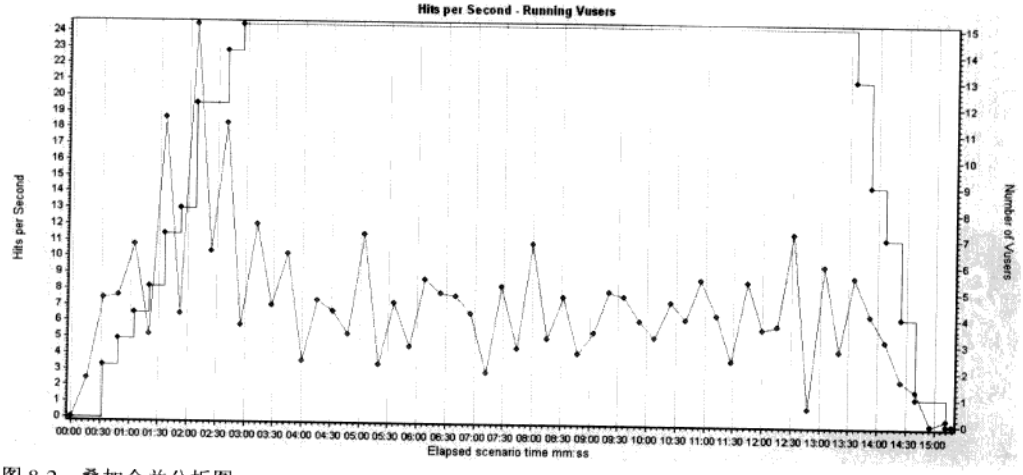


图 8-2 叠加合并分析图

- 平铺：在平铺布局查看，共用同一个 X 轴，合并进来的图显示在当前图的上方，如图 8-3 所示。

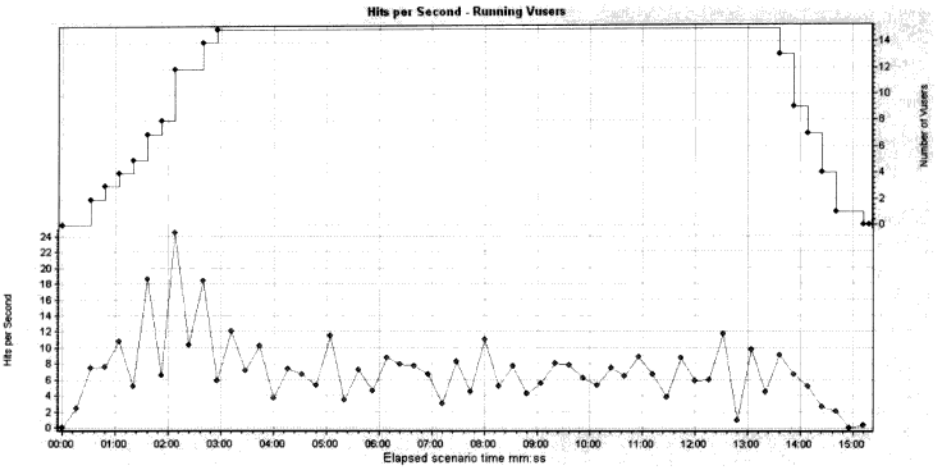


图 8-3　平铺合并分析图

- 关联：合并后当前活动图的 Y 轴变为合并图的 X 轴，被合并图的 Y 轴作为合并图的 Y 轴，如图 8-4 所示。

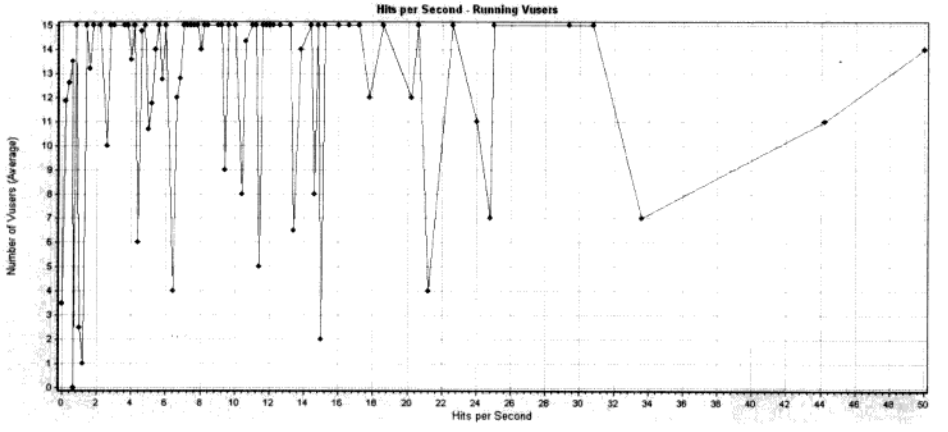


图 8-4　关联合并分析图

3）合并图标题：设置视图合并后的标题。

8.1.2　实例讲解

上面讲述了分析图合并的原理，下面通过一个实例来分析如何对数据图进行合并分析，该实例是将“正在运行的 Vuser”图、“每秒点击数”图和“吞吐量”图三个图进行叠加合并图分析。这里是将三个图进行合并，和前面讲述的对两个分析图进行合并的原理是一致的。

合并后，如图 8-5 所示。

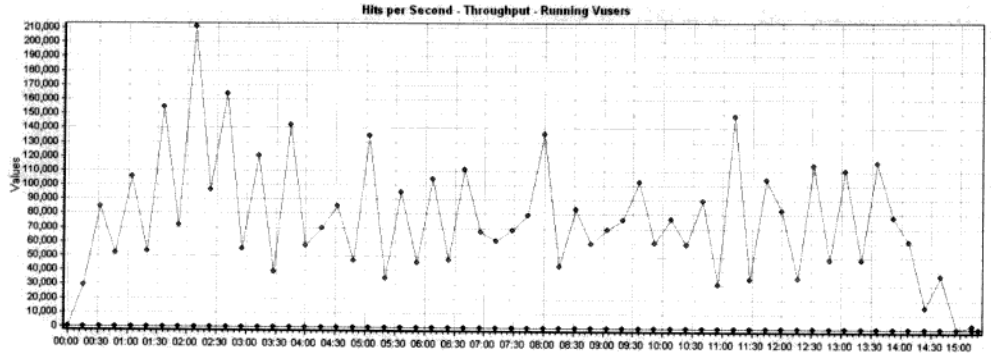


图 8-5 合并后的分析图

这时发现“正在运行的 Vuser”图和“每秒点击数”图几乎看不到。在实际测试过程中可能经常遇到这样的情况，这是因为 Y 轴的粒度太小影响分析，有时 X 轴的粒度太小，也会影响分析，这时就要调整 X 轴的粒度或 Y 轴的显示比例。在该实例中只要调整 Y 轴的显示比例就可以。这里将“正在运行的 Vuser”和“每秒点击数”两个视图的 Y 轴放大 10000 倍，通过 Configure Measurements 设置来更改。更改 Y 轴显示比例后，如图 8-6 所示。

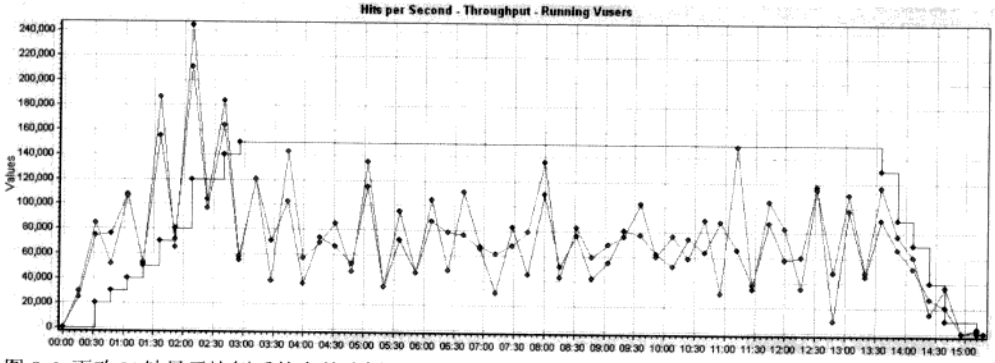


图 8-6 更改 Y 轴显示比例后的合并分析图

有时候可以对分析图进行筛选设置，但在该实例中可不用进行这方面的设置。

下面是分析合并图常用的三个步骤。合并完成之后要对这个合并图，以及这三个图的趋势进行分析。主要是分析这三种图的趋势是否正确。

首先，找到影响几个图变化趋势的决定因子。在这里先抽出“正在运行的 Vuser”图来分析，因为其他两个图的变化与 Vuser 用户有关。

接着，应该了解合并图中其他的图与该决定因子的关系。这里“每秒点击数”图和“吞吐量”

图与“正在运行的 Vuser”图是成正比的关系。也就是说随着 Vuser 用户的增加，每秒点击数与吞吐量都增加。

再次，分析决定因子图的变化趋势。“正在运行的 Vuser”图的变化趋势是先加载 Vuser 用户，当全部加载完成后，所有的 Vuser 用户会运行一段时间，再开始释放 Vuser 用户。

最后，通过各图之间的关系来判断其他的这些图变化的趋势是否正确。最后判断“每秒点击数”图和“吞吐量”图变化趋势，要判断这两个图变化趋势是否与“正在运行的 Vuser”图变化趋势一致。如果一致，则说明结果分析图是正确的；否则就说明结果是不正确的，如果发现有异常的现象可以再借助其他的分析方法来确定真实的原因。

8.2 分析图关联

8.2.1 分析图关联原理

在当前的分析图中点击右键，在右键菜单中选择 Auto Correlate，将打开 Auto Correlate 对话框，如图 8-7 所示。图中显示的数据范围为选定的度量，如果不手动更改度量范围，那么会显示出整个运行结果的度量。

“Time Range（时间范围）”选项卡用来设置分析关联图度量的时间范围。

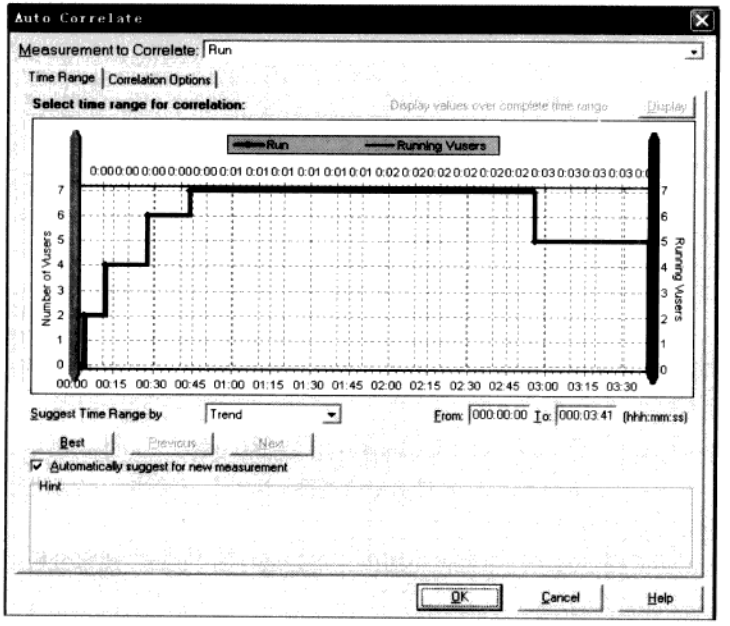


图 8-7 关联图时间范围

在 Suggest Time Range by 下拉列表框中可以看到有两种时间范围方式：Trend（趋势）和 Feature（功能），其实对于时间范围方式有三种，Suggest Time Range by 下拉列表框下面还有一个 Best 按钮。这三种方式的含义如下：

- Trend（趋势）：选择关联度量值变化趋势相对稳定的一段为时间范围。
- Feature（功能）：在关联度量值变化相对稳定的时间内，选择一段大体与整个趋势相似的时间范围。
- Best（最佳）：选择关联度量值发生明显变化趋势的一段时间范围。

也可以手动调整时间范围，具体有两种方式，一种是手动填写具体的开始时间和结束时间；另一种是拖动绿色和红色线来指定起止时间，其中绿色线表示起始时间，红色线表示终止时间。

通过“Correlation Options（自动关联）”选项卡可以设置要关联的图、数据间隔和输出选项，如图 8-8 所示。

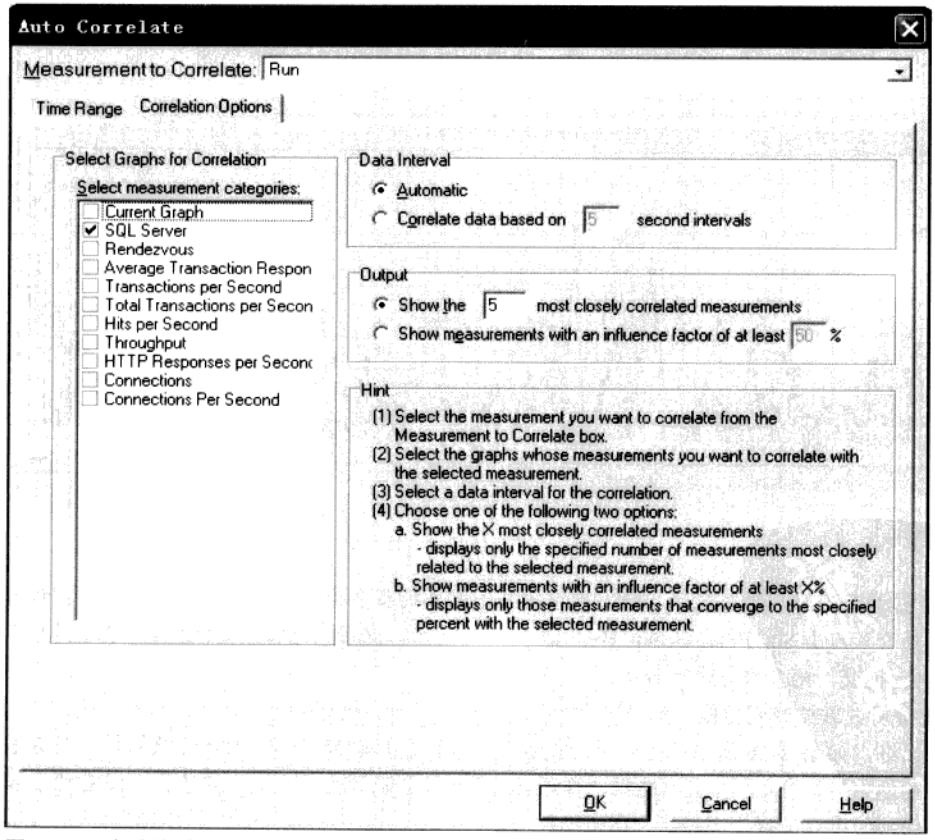


图 8-8 “自动关联”选项卡

在 Select Graphs for Correlation 中选择需要关联的图。在 Data Interval 组合框中选择计算关联度量轮询之间的时间间隔，可以设置为自动，也可以自定义。在 Output 组合框中选择显示输出的级别。

8.2.2 实例讲解

分析图关联是通过复杂的统计学方法，精确定位哪些因素对交易响应时间的影响最大，关联并不关注具体的数据，而是关注于参数样本在特定时间范围内的状态、趋势。只有折线图可以使用 Auto Correlation（除 Web Page Diagnostic 折线图外）。

实例：分析“平均事务响应时间”图与“Windows 资源”图关联的情况。

1）选择要关联的图为当前活动图，实例中的当前活动图应该为“平均事务响应时间”图，再选择被关联图，被关联图为“Windows 资源”图。在“平均事务响应时间”图中点击右键，选择 Auto Correlation，在弹出的 Auto Correlation 对话框中选择 Correlation Options 选项卡，在其中选择需要关联的图，这里选择“Windows 资源”图。生成如图 8-9 所示的自动关联图。

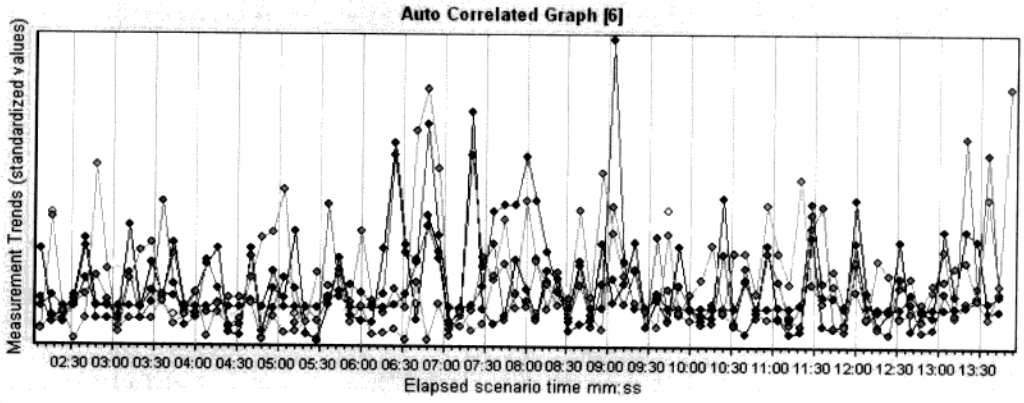


图 8-9 初始关联图

2）设置过滤条件。自动关联后，会发现很多并不需要的事务也被关联进来了，这时就需要对其进行过滤处理。点击 Set Filter/Group By，如图 8-10 所示，这里只选择“商业机会_进入界面”和“商业机会_提交”两个事务进行分析。

3）设置分析关联的时间范围。在过滤后的关联图中，点击右键选择 Auto Correlation，弹出“自动关联”对话框，如图 8-11 所示。可以手动设置关联分析的时间范围，但需要注意的是，在这里要选择波折的地方进行分析，并且不能只选择只有一个波折的时间范围，至少要选择一段有两个以上波折的曲线。如果只选择一个波折，在自动关联后，会发现很多项的关联度都为 100。这个分析就没有意思，因为只选择一个波折的时间范围太小。

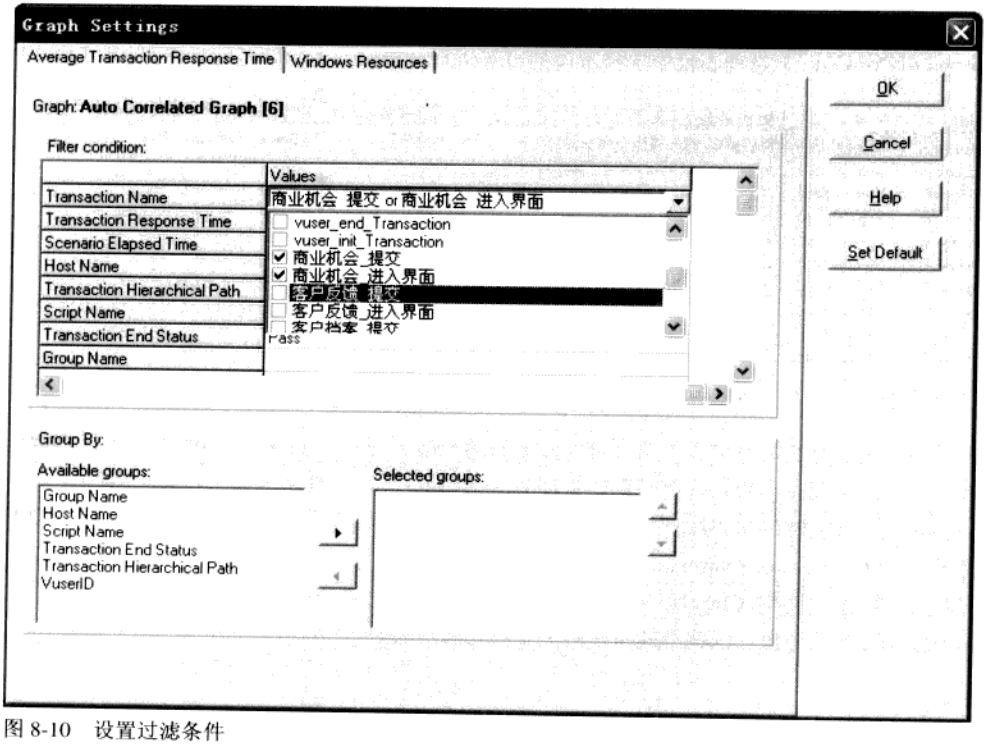


图 8-10 设置过滤条件

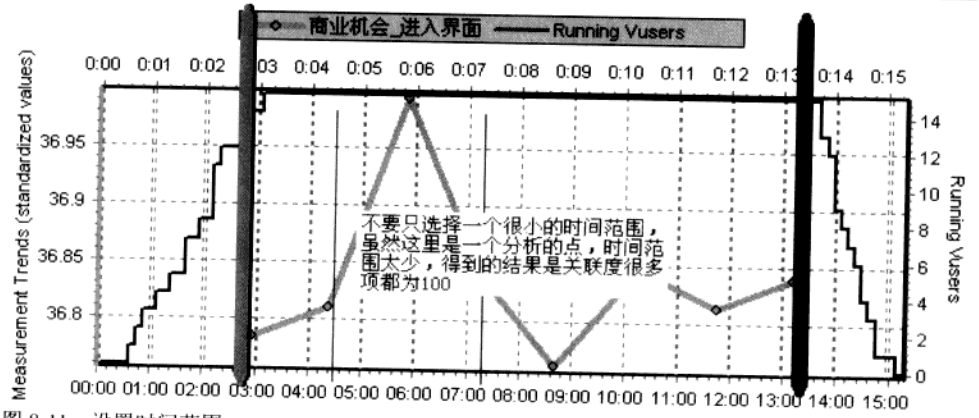


图 8-11 设置时间范围

4）分析关联度。自动关联后会看到下面列出所有关联度的信息，如图 8-12 所示。现在选择关联度最高的来分析。这里关联度最高的为 Interrupts/sec，关联度为 62。接下来到“Windows 资源”

图中查看 Interrupts/sec 项变化的情况，来确定系统资源中这一项是否可能存在瓶颈。在关联图上方 Measurement to Correlate 下拉列表框中选择不同的事务进行分析。

Color	Graph	Scale	Measurement	Correlation Match	Correlation	Machine Na...	Mc
☑	Average Tra...	Standardized	商业机会_进入界面	100	Directly Related	N/A	N/
☑	Windows R...	Standardized	Interrupts/sec (Processor _Total):127.0.0.1	62	Inversely Related	127.0.0.1	
☑	Windows R...	Standardized	% Processor Time (Processor _Total):127.0.0.1	51	Inversely Related	127.0.0.1	
☑	Windows R...	Standardized	Pages/sec (Memory):127.0.0.1	48	Directly Related	127.0.0.1	
☑	Windows R...	Standardized	Private Bytes (Process _Total):127.0.0.1	45	Inversely Related	127.0.0.1	
☑	Windows R...	Standardized	% Disk Time (PhysicalDisk _Total):127.0.0.1	43	Directly Related	127.0.0.1	
☐	Windows R...	Standardized	Page Faults/sec (Memory):127.0.0.1	38	Inversely Related	127.0.0.1	
☐	Windows R...	Standardized	Processor Queue Length (System):127.0.0.1	34	Inversely Related	127.0.0.1	
☐	Windows R...	Standardized	Pool Nonpaged Bytes (Memory):127.0.0.1	30	Directly Related	127.0.0.1	
☐	Windows R...	Standardized	File Data Operations/sec (System):127.0.0.1	27	Inversely Related	127.0.0.1	

图 8-12　分析关联度

到这里整个关联分析就结束了。通过合并和关联的实战，可以看出 Atuo Correlation 与 Merge 存在一些共同点，但同时也存在一些区别，具体的区别如下：

1）Merge 不能选择特定的时间进行切片，所以只有先用 Merge 看整体趋势、分析全局。找到恰当的位置后，再使用 Auto Correlation 切片，进一步分析。

2）Merge 的输出没有 Correlation Match 这个值，即使使用 Merge 的 Correlate 选项也没有 Correlation Match 这个值，也就不能衡量两个参数之间的关系。

8.3　页面细分

8.3.1　页面细分原理

在平均事务图中点击右键选择 Show Transaction Breakdown Tree，生成 Web Page Diagnostics 图。通过分解页面可以发现，页面中哪些组件响应时间较大？平均事务响应时间过长是由服务器还是由网络环境引起的？

正常的从浏览器发送一个请求到最后显示，整个过程由以下时间片组成，如图 8-13 所示。

图 8-13　网络时间解析图

1）浏览器向服务器发送一个请求，一般情况下，客户端的请求首先被发送到 DNS 服务器上，通过域名解析，将 DNS 名解析为 IP 地址。其中域名解析的时间就是 DNS 解析的时间（DNS Resolution）。通过这个时间可以确定 DNS 服务器或 DNS 服务器的配置是否有问题。如果 DNS 服务器运行情况良好，这个时间会比较小。

2）DNS 解析完成后，请求被送到 Web 服务器，之后浏览器与 Web 服务器之间需要建立一个

初始化连接。建立连接的过程就是连接时间（Connection）。这样通过这个时间就可以判断网络的情况，也可以判断 Web 服务器是否能够响应这个请求。如果正常，这个时间会比较小。

3）建立连接后，Web 服务器发出第一个数据包，经过网络传输到客户端，浏览器成功接收到第一个字节的时间就是 First Buffer 的时间。这个度量时间不仅可以表示 Web 服务器的延迟时间，还可以表示网络反应时间。

4）从浏览器接收到第一个字节起，直到所有的字节都成功接收为止。这个度量可以判断网络的质量（可以用 size/time 比来计算接收速率），其他的时间还有 SSL Handshaking（SSL 握手协议，用到该协议的页面比较少）、Client Time（请求在客户端浏览器延迟时间，可能是由于客户端浏览器的 ThinkTime 或者客户端其他方面引起的延迟）、Error Time（从发送一个 HTTP 请求，到 Web 服务器发送回一个 HTTP 错误信息所需要的时间）。

页面细分的具体内容如下：

切换到 Breakdown Tree，这里显示了事务包含的所有页面，如图 8-14 所示。显示了事务“商业机会_提交”包含的所有页面。

图 8-14 事务包含页面图

在 Legend 框中，可以选择需要的页面进行分析，Web Page Diagnostics 中显示了该页面运行时的响应时间，Diagnostics options 中显示了该页面包含的所有组件，以及组件的大小和组件下载的时间，如图 8-15 所示。

选择 Component（Over Time），这里显示了各组件在场景运行过程中下载的时间，如图 8-16 所示。可以通过 按钮和 按钮来切换是只显示选中组件的下载时间还是显示所有组件的下载时间。

选择 Download Time（Over Time），可以看到在场景运行时，组件在网络传输过程中的各部分时间，如图 8-17 所示。

为了确定问题是由服务器还是由网络引起的，选择 Time to First Buffer（Over Time），如图 8-18 所示，该图显示了在网络传输过程中和服务器两部分分别使用的时间。

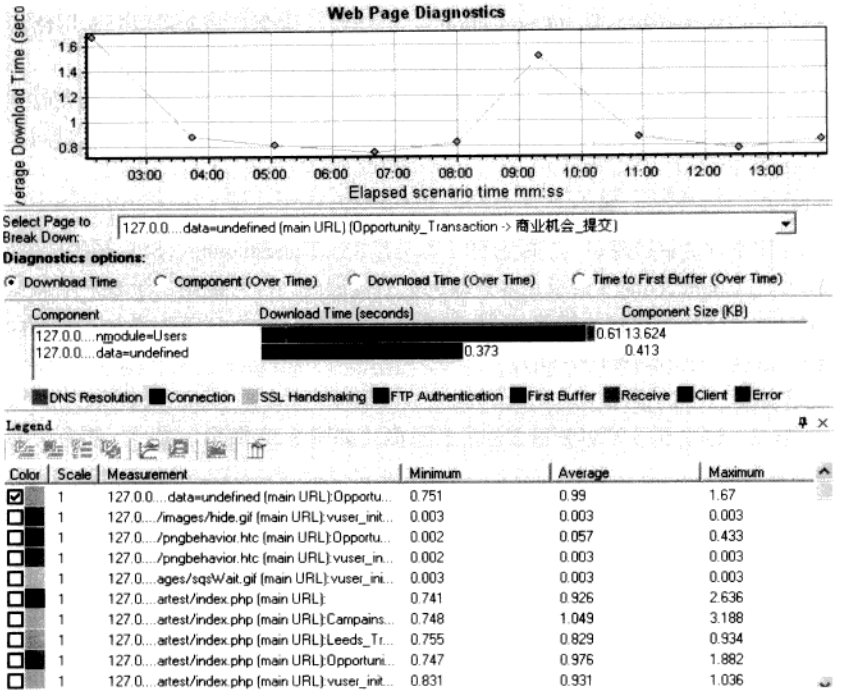


图 8-15 Download 时间图

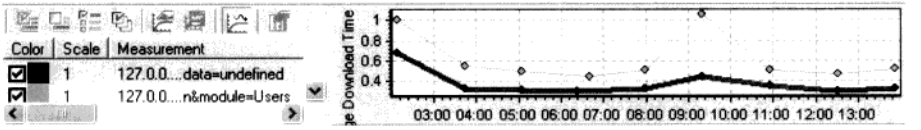


图 8-16 各组件下载时间图

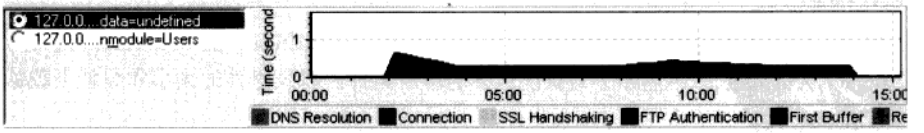


图 8-17 各组件运行时间图

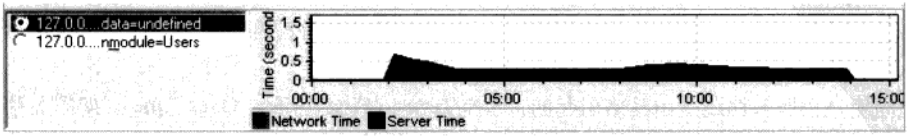


图 8-18 First Buffer 时间图

8.3.2 实例讲解

页面细分技术主要用来分析失败事务是由哪些组件引起的，对于成功的事务就没必要。下面通过一个实例来了解页面细分技术的一般步骤：

1）打开“平均事务响应时间”图，打开 Graph Settings 对话框，对过滤条件进行设置，将 Transaction End Status 设置为 Fail，Transaction Name 设置为要分析的事务即可，如图 8-19 所示。

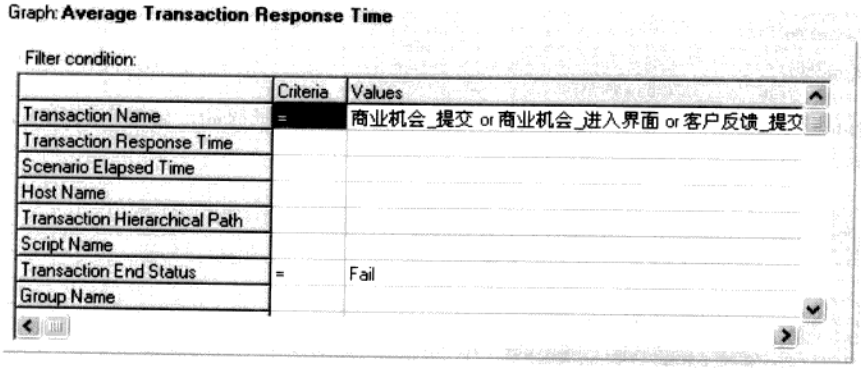


图 8-19 设置过滤条件

2）显示事务细分树。在“平均事务响应时间”图中点击右键，选择 Show Transaction Breakdown Tree。此时左下角弹出一个 Breakdown Tree 图，如图 8-20 所示。这里显示了所有失败事务图。

图 8-20 Breakdown Tree 图

3）显示页面细分图。选择需要分析的事务，这里选择“营销活动_提交”事务进行分析。点击右键，选择“Web Page Diagnostics for ‘营销活动_提交’”，如图 8-21 所示。Analysis 分析器生成 Web Page Diagnostics 图。

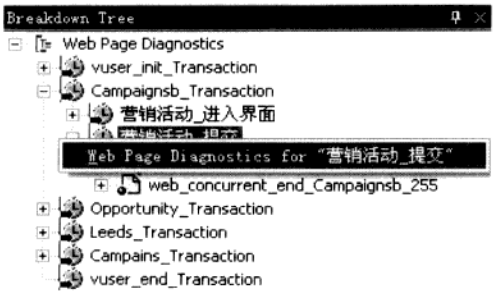


图 8-21 生成页面细分图

4）查看 Download Time 图。选择 Download Time 查看各组件所花费的时间，如图 8-22 所示，第一个组件所花费的时间最长并且还包含错误信息。

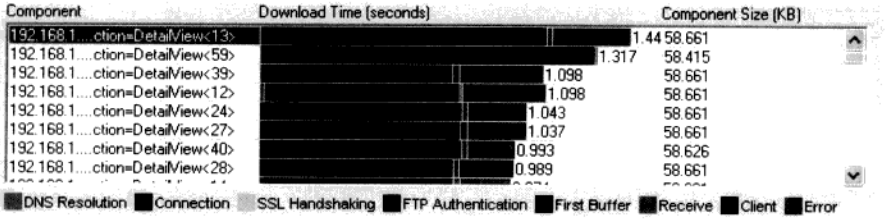


图 8-22 查看 Download Time 图

5）手动查看该组件响应时间。选中该组件，点击右键，选择 Copy the full path to the clipboard，将路径拷贝到 IE 浏览器中进行预览。或选择 View page browser 直接打开该页面进行预览。手动预览能判断该页面响应的真实时间。如果手动预览该页面和测试的结果一致，说明事务失败确实是由于该页面响应时间引起。如果不是手动预览响应很快，那么要进一步判断是由测试环境引起还是由网络引起。

6）查看 Download Time（Over Time）图。如图 8-23 所示，图中详细地记录了请求在各阶段所花费的时间，通过 Download Time（Over Time）图可以看出 DNS 服务器花在解析域名的时间在 1 秒以上，说明 DNS 服务器可能有问题。

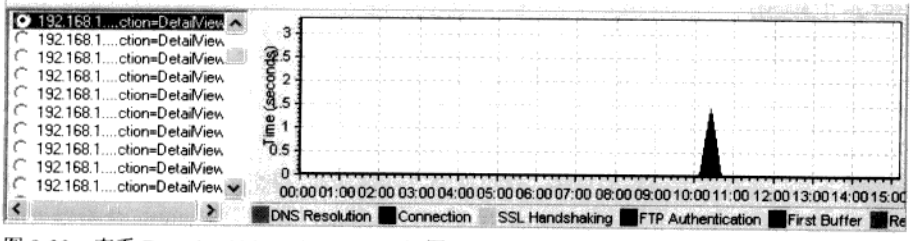


图 8-23 查看 Download Time（Over Time）图

7）查看 Time to First Buffer（Over Time）图。通过该图观察问题到底是由服务器引起还是由网络引起，如图 8-24 所示。这里发现服务器的时间明显超过 1 秒，而网络时间不到 0.5 秒，说明测试的网络环境是没有问题的，问题主要出在服务器。

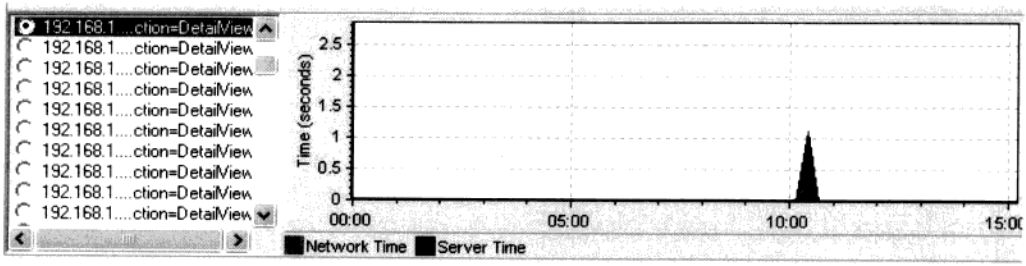


图 8-24 查看 Time to First Buffer（Over Time）图

整个分析过程已经结束，通过分析发现“营销活动_提交”事务失败的原因来自服务器，DNS 服务器和 Web 服务器都有可能是引起问题的原因。

8.4 钻取技术

8.4.1 钻取技术原理

在当前活动图中，点击右键，选择 Drill Down 可以对当前图进行钻取。通过钻取技术可以获取事务图更深层次的资料。能够钻取的组资料由 Drill Down Options 对话框中的 Group By 决定，但一般都会包含：Host Name（主机名）、Vuser ID（虚拟用户 ID）等信息，这是由当前活动的图所决定的，对于不同的事务图其 Group By 中的信息有所不同。

点击右键，选择 Drill Down 弹出 Drill Down Options 对话框，如图 8-25 所示。在这里可以设置要钻取的事务图和组信息。钻取之后，会按 Group By 中选择的钻取组的信息进行排序，并且组是以不同的曲线来显示，如果钻取后有两个 Vuser ID，就会显示出两条不同的曲线。

钻取技术通常有以下几个特点：

1）在一个活动图中，选择一个需要的组进行显示，这时钻取技术可以帮助进行特定的测量。

2）组信息由当前活动图所决定，对于不同的图，组信息有所不同。

3）可以钻取每个 Vuser 的平均事务响应时间，并可以按 Vuser ID 进行排序。

4）钻取后的信息会按组中不同的元素与不同的曲线显示出来，如不同的 Vuser ID 显示不同的曲线图。

钻取技术不适用页面细分图。

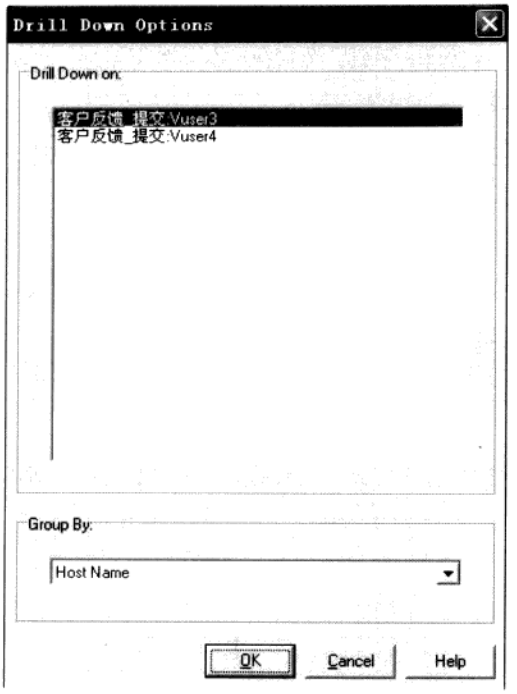


图 8-25 钻取属性设置

8.4.2 实例讲解

实例：钻取测试结果中哪些 Vuser ID 执行事务失败。

1）对“平均事务响应时间”图进行过滤，过滤出失败的事务图。有一些图不需要过滤直接用即可，如“每秒点击率”图。

2）过滤后点击右键，选择 Drill Down，在弹出的 Drill Down Options 对话框中，选择要钻取的事务和组信息，这里选择“客户反馈_提交”事务和按 Vuser ID 进行钻取。

3）这时会显示出所有 Vuser ID 对应的信息，如图 8-26 所示。

Color	Scale	Measurement	Graph's Mini...	Graph's Ave...	Graph's Max...	Graph's Me...
☑	1	客户反馈_提交:Vuser3	24.544	25.216	30.082	24.719
☑	1	客户反馈_提交:Vuser4	24.547	24.685	25.017	24.657

图 8-26 钻取失败的 Vuser ID 信息

4）钻取后，可以选择不同的组，对当前钻取的结果再次进行钻取，得到更多的信息。

5）或者在左下角 Properties 对话框中的 Group By 框中点击…，弹出 Graph Settings 对话框，如图 8-27 所示。在该对话框中可以同时设置多个 Group By 条件。

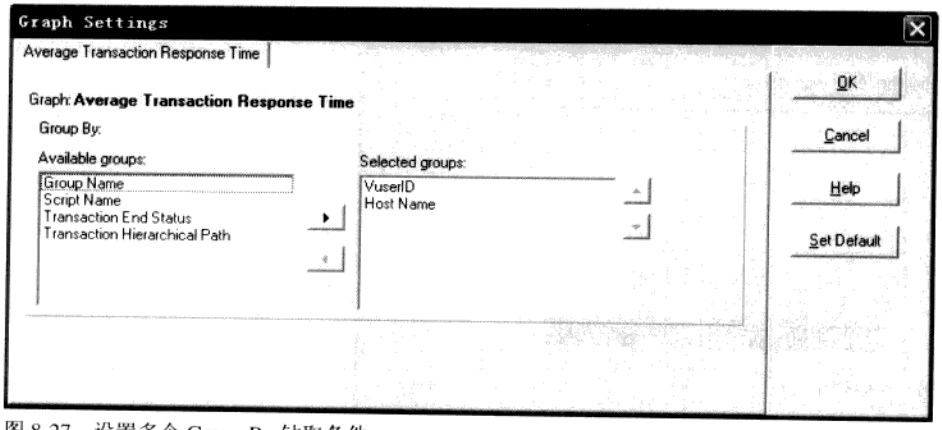


图 8-27 设置多个 Group By 钻取条件

整个钻取过程已经结束，钻取后应该借助其他的手段来帮助进行更深层次的分析，才能找到系统真正的瓶颈。

8.5 导入外部数据

通过 LoadRunner Analysis 导入数据，可以将非 Mercury Interactive 数据导入并集成到 LoadRunner Analysis 会话中。完成导入操作后，可以使用 Analysis 工具的所有功能以图的形式查看会话中的数据文件。

假如一个 NT 性能监视器在服务器上运行，并对其行为进行度量。在服务器上执行 LoadRunner 方案后，可以检索性能监视器的结果，并将数据集成到 LoadRunner 的结果中。这样能够将两数据集的趋势和关系相关联。

8.5.1 导入数据工具

LoadRunner 自带了一个导入数据的工具，选择 Tools→External Monitors→Import Data，弹出 Import Data 对话框，如图 8-28 所示。

- File Format：选择导入外部文件的格式。
- Add File：选择要导入的文件。
- Date Format：选择日期格式。
- Time Zone：选择时区。
- Machine Name：计算机名称。这里除了 NT Performance Monitor(*.csv)和 Win 2K Performance Monitor(*.csv)两种文件格式不需要填写机器名称外，其他的文件格式都必须填写机器名称。同时该设置会将计算机名与度量相关联。例如，设置计算机名为“arivn”，一个度量为“File IO rate”，将会命名为“File IO rate:arivn”。

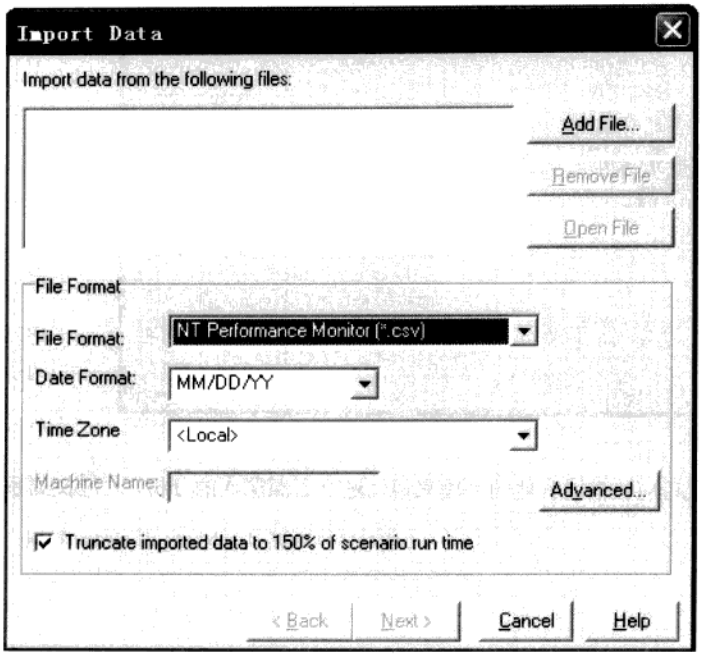


图 8-28 导入数据

如果要指定字符分隔符和符号，请点击 Advanced... 按钮，如图 8-29 所示。

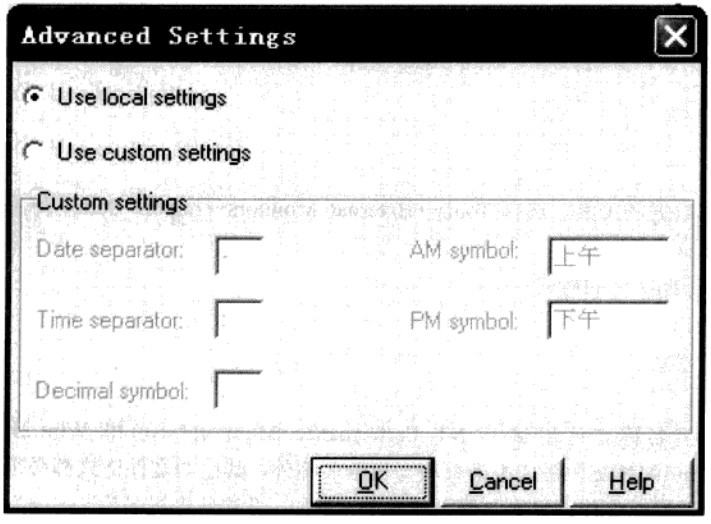


图 8-29 高级设置

这里可以选择 Use local settings 和 Use custom settings 两种：

Use local settings：保存默认的区域设置。禁止对话框中的 Use custom settings 区域。

Use custom settings：定义自己的设置。启动对话框中的 Use custom settings 区域。

- Date separator：日期分隔符，输入一个自定义符号，例如 11-10-03 中的“-”。
- Time separator：时间分隔符，输入一个自定义符号，例如 10:54:29 中的冒号“:”。
- Decimal symbol：小数点，输入一个自定义符号，例如 5.5 中的小数点“.”。
- AM symbol：AM 符号，输入一个自定义符号，以标识午夜 0 点至中午 12 点之间的时段。
- PM symbol：PM 符号，输入一个自定义符号，以标识中午 12 点至午夜 0 点之间的时段。

设置完成之后，点击“下一步”按钮，如图 8-30 所示，选择监视器类型。

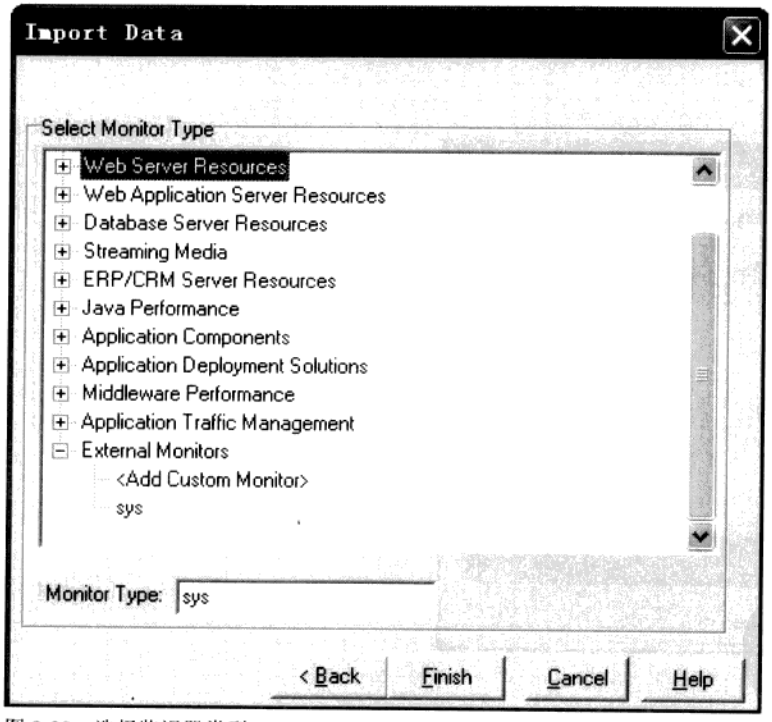


图 8-30 选择监视器类型

如果监视器类型列表不包含需求的监视器，那么可以自定义一个新的监视器类型。选择 External Monitors→Add Custom Monitor，弹出 Add Custom Monitor 对话框，如图 8-31 所示。

在这里输入监视器名称和相关描述，点击“确定”按钮即可。之后在选择监视器类型对话框中点击“完成”按钮即可。

LoadRunner 会显示出导入数据的过程和结束的状态，如图 8-32 所示。

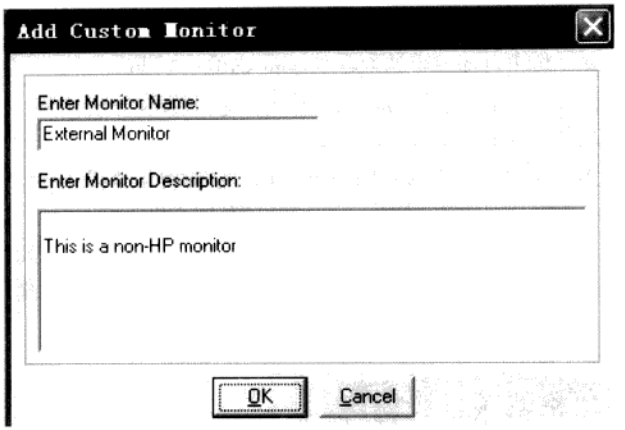


图 8-31 添加自定义监视器

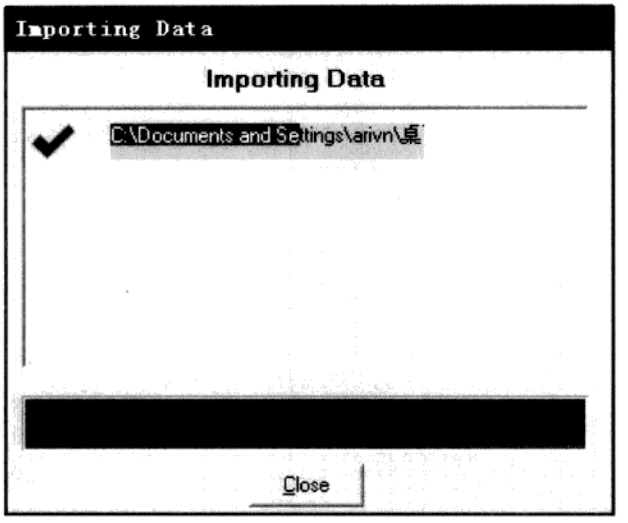


图 8-32 数据导入后的状态

8.5.2 自定义文件格式

LoadRunner 支持下列文件类型：

- NT Performance Monitor(*.csv)（NT 性能监视器）
- Win 2K Performance Monitor(*.csv)（Windows 2000 性能监视器）
- Standard Comma Separated Files (*.csv)（标准逗号分隔文件）

- Standard Microsoft Excel Files (*.csv)（主从逗号分隔文件）
- Master-Detail Comma Separated Files(*.csv)（Microsoft Excel 文件）
- Master-Detail Microsoft Excel Files(*.csv)（主从 Microsoft Excel 文件）
- XLS

当导入的数据为 Analysis 不支持的外部数据文件时，可以定义一个新的文件格式。在导入数据对话框的文件格式中选择 Custom File Format，弹出如图 8-33 所示的定义外部格式的对话框。

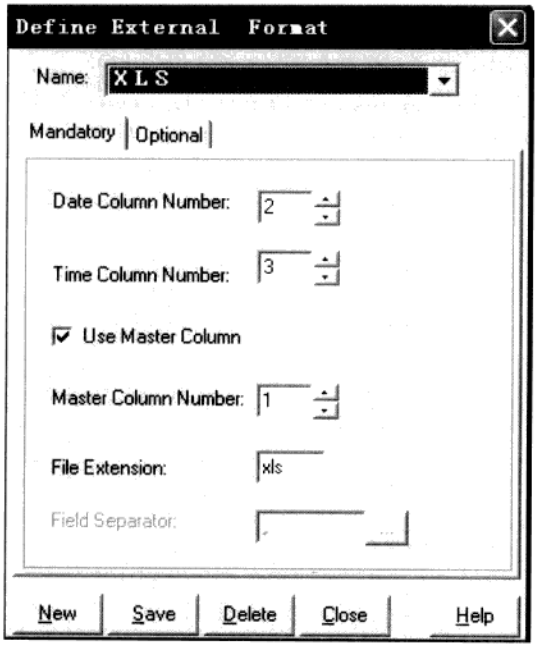


图 8-33 定义外部格式

点击“新建”按钮，在弹出的输入新的格式名对话框中，输入新建格式的名称，如 My_Format。在这里有两个选项卡“Mandatory（强制）”和“Optional（可选）”。

Mandatory 选项卡的内容如下：

- Date Column Number（日期列编号）：输入包含日期的列。如果存在主列，请指定其编号。
- Time Column Number（时间列编号）：输入包含时间的列。
- Use Master Column（使用主列）：如果数据文件包含主列，请选择该选项。主列指定该行对某一常规测量的特定细分。
- File Extension（文件扩展名）：输入文件后缀名。
- Filed Separator（字段分隔符）：输入用于分隔行中各字段的字符。要选择字段分隔符，请点击“浏览”按钮，在“定义字段分隔符”对话框中选择一个字符。

Optional 选项卡内容如下：

- Date Format（日期格式）：指定导入数据文件中的日期格式。
- Time Zone（时区）：选择记录外部数据文件所在的时区。一般的设置为本地时区。
- Machine Name（机器名称）：指定运行监视器的计算机名。该设置将机器名与度量相关联。
- Exclude Columns（排除列）：指明在导入数据时要排除的列，如包含描述性注释的列。如果排除多列，要用逗号分隔列表指定这些列。
- Convert file from UNIX to DOS format（将文件格式从 UNIX 转换为 DOS）：监视器通常在 UNIX 计算机上运行。选中该选项，将数据文件格式转换为 Windows 格式。在 UNIX 文件中，所有换行符（ASCII 码为 10）之后均附加了一个回车符（ASCII 码为 13）。
- Skip the first *** lines（跳过前***行）：指定在读取数据之前要忽略的文件开始处的行数。通常情况下文件的前几行包含标题和子标题。

9 特殊协议

9.1 Windows Sockets（WinSock）协议

WinSock 协议是一个底层协议。所有的高级协议（如 FTP、HTTP 协议等），以及所有基于 Windows 的应用（如 IE、FTP），其底层通信都是使用 WinSock 协议，因此任何高级协议的底层都使用 WinSock 通信。那么什么时候才选择 WinSock 协议呢？如果在录制脚本前，找不到更合适的协议时，都可以选择 WinSock 协议进行录制脚本。WinSock 的另一个特点就是非常适合应用程序代码级，所以需要查看缓冲区发送和接收的实际数据时，也可以选择 WinSock 协议。

使用 WinSock 协议开发脚本的过程如下，如图 9-1 所示。

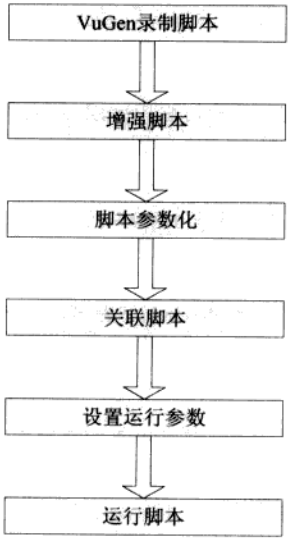


图 9-1 WinSock 协议脚本开发过程

9.1.1 Windows Sockets 录制选项设置

选择 Tools→Recording Options 或在 Start Recording 对话框中选择 Options，都弹出如图 9-2 所示的 Recording Options 对话框。

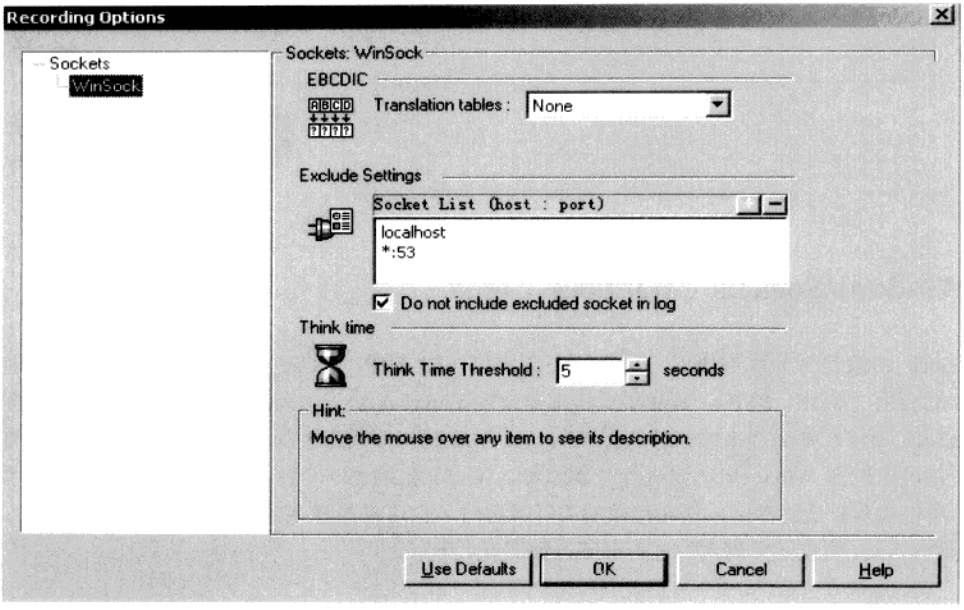
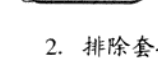


图 9-2 WinSock 选项对话框

1. 配置转换表

需要以 EBCDIC 格式显示数据，请在录制选项中指定转换表。通过转换表可以指定录制会话的格式。这适用于大型机或 AS/400 服务器上运行的用户。服务器和客户端计算机都从系统上所安装的转换表中确定数据的格式。可以在 Translation tables 下拉列表框中选择转换选项。其中前四位表示服务器的格式，后四位表示客户端的格式。例如，002504e4 表示服务器格式为 0025，客户端格式为 04e4。



如果数据是 ASCII 格式，则不需要转换。此时必须选择默认的 None。如果确实选择了转换表，则 VuGen 将转换该 ASCII 数据。

2. 排除套接字

VuGen 支持“排除套接字”功能，可以从录制会话中排除特定的套接字。要从脚本中排除某个套接字的所有操作，请在“排除套接字”列表中指定该套接字的地址。要向列表中添加套接字，点击该框右上角的加号，然后以下列格式之一输入套接字地址，如表 9-1 所示。

表 9-1 排除套接字表

值	含义
主机:端口	仅排除指定主机上的指定端口
主机	排除指定主机上的所有端口
:端口	排除本地主机上的指定端口号
*:端口	排除所有主机上的指定端口号

这样可以将多个主机或端口添加到排除套接字列表中。要从排除列表中删除套接字，请选择该套接字地址，然后点击该框右上角的减号。

默认情况下，VuGen 不记录在“已排除套接字列表”中的套接字的操作，如果需要指示 VuGen 记录已排除的套接字操作，请清除 Do not include excluded socket in log 复选框。如果已排除的套接字启用了日志记录，那么在日志文件中，这些套接字的操作之前会加上 Exclude 字样。

3. 设置思考时间阈值

在录制期间，VuGen 会自动插入操作者的思考时间。可以设置阈值级别，录制的思考时间如果低于此阈值，则被忽略。如果录制的思考时间超出了阈值级别，VuGen 将在 LRS 函数之前放置 lr_think_time 语句。如果录制的思考时间低于阈值级别，则不会生成 lr_think_time 语句。

9.1.2 Windows Sockets 录制

前面已经讲述了使用 Windows Sockets 协议开发的过程，下面结合实例对这个开发过程进行详细讲解。

1. 录制脚本

首先录制一段脚本，被录制的程序主要功能是通过网络将数据从一台机器传输到另一台机器。下面是录制好的代码，后面所有关于这一章节的讲解都是围绕这段脚本来进行。

```
Action()
{
    lrs_create_socket("socket0", "TCP", "LocalHost=0", "RemoteHost=58.222.18.90:80", LrsLastArg);
    lrs_send("socket0", "buf0", LrsLastArg);
    lrs_receive("socket0", "buf1", LrsLastArg);
    lr_think_time(8);
    lrs_create_socket("socket1", "UDP", "LocalHost=8080", LrsLastArg);
    lrs_send("socket1", "buf2", "TargetSocket=edan-3a89b690e0:8080", LrsLastArg);
    lrs_receive("socket1", "buf3", LrsLastArg);
    return 0;
}
```

data.ws 文件中的内容如下：

```
;WSRData 2 1
send buf0 201
    "GET /download/cmsoft/UpgradeInfo/NetAssist.inf HTTP/1.1\r\n"
```

```
        "If-Modified-Since: Fri, 03 Jul 2009 03:57:53 GMT\r\n"
        "If-None-Match: \"e8ffa77092fbc91:633a6\"\r\n"
        "User-Agent: Internet+Explorer\r\n"
        "Host: www.cmsoft.cn\r\n"
        "\r\n"
    recv buf1 297
        "HTTP/1.1 304 Not Modified\r\n"
        "Date: Tue, 14 Jul 2009 12:25:16 GMT\r\n"
        "Content-Location: http://www.cmsoft.cn/download/cmsoft/UpgradeInfo/NetAssi"
        "st.inf\r\n"
        "Last-Modified: Fri, 03 Jul 2009 03:57:53 GMT\r\n"
        "Accept-Ranges: bytes\r\n"
        "ETag: \"e8ffa77092fbc91:<port1>\"\r\n"
        "Server: Microsoft-IIS/6.0\r\n"
        "X-Powered-By: ASP.NET\r\n"
        "\r\n"
    send buf2 5
        "hello"
    recv buf3 5
        "hello"
    -1
```

2. 增强脚本

增强脚本部分主要是对脚本进行插入开始事务、结束事务和集合点的操作，也包括对脚本进行单机调试，保证脚本能正确运行。

3. 脚本参数化

增强脚本之后，需要对脚本进行参数化。首先要确定脚本中哪些部分需要进行参数化。上一实例的脚本中 buf2 是手动输入的要发送的内容，buf3 是接收 buf2 的内容，两者内容一致，这里可以对发送的 buf2 中的数据进行参数化。但需要注意的问题是，由于在 Windows Sockets 协议录制时所有的传输数据均保存在数据文件 data.ws 中，因此在进行参数化时主要是对该文件中的相关数据进行参数化。在 data.ws 中选择 buf2 中内容“hello”，点击右键→Replace with a Parameter，如图 9-3 所示。接下来的参数化过程就和普通的参数化过程一致。

那么为什么不需要对 buf3 进行参数化呢？可以分析一下 buf2 和 buf3 的数据，这两个缓存中的数据内容是一样的，buf2 的关键字是 send，buf3 的关键字是 rev，也就是说 buf2 的内容是要传送给远程服务器的内容，而 buf3 的内容是录制时从远程服务器返回的数据，当然这个数据也是希望返回的数据。因此，一般情况下只需要对 send 中的相关内容进行参数化。

这里只是需要参数化 data.ws 数据文件中的内容，那么如果希望传输一个文件中所有的内容应该怎么办呢？这是在编写脚本时经常会遇到的一个问题，而 LoadRunner 本身并未提供直接读取文件的方法，不过 LoadRunner 可以兼容 C 语言，这样就可以使用 C 语言来增强这段代码。如下面的代码，让传输 buf2 的内容变成传输一个文件的内容。

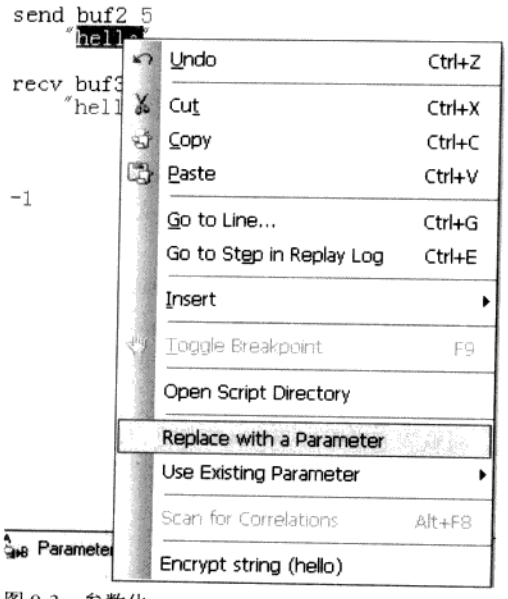


图 9-3 参数化

```
int count;
char userbuffer[500];
long filestream;
char * filename = "E:\\1.txt";
if ((filestream = fopen(filename, "r")) == NULL ) {
    lr_error_message ("Cannot open %s", filename);
    return -1;
    }
      // Read until end of file
      while (!feof(filestream)) {
      // Read 500 bytes while maintaining a running count
            count = fread(userbuffer, sizeof(char), 500, filestream);
            lr_output_message ("%3d bytes read", count);
            if (ferror(filestream)) { /* Check for file I/O errors */
                  lr_output_message ("Error reading file %s", filename);
                  break;
            }
      }
      if (fclose(filestream)) // Close the file stream
            lr_error_message ("Error closing file %s", filename);
lrs_set_send_buffer("socket1",userbuffer,count);
```

上面这两种参数化的方法是最常用的方法。

4. 关联脚本

参数化完成之后，进行脚本回放，发现回放的日志文件中提示下面的信息：

Action.c(25): Mismatch in buffer's length (expected 299 bytes, 986 bytes actually received, difference in 687 bytes)。

这说明录制与回放的内容没有匹配，回放日志文件中详细记载了 Expected Buffer 和 Received Buffer 的值，通过比较这两种值可以发现录制和回放过程中不一致的地方。回放日志文件中提示缓存长度不匹配。

为什么脚本回放会提示缓存长度不匹配？首先来了解 Mismatch 匹配的机制。Mismatch 有两种匹配方式：长度匹配和内容匹配。所谓长度匹配是当 lrs_receive 在接收到数据之后就会和期望的缓冲区数据进行长度（字符数）对比，如果实际接收到的数据长度不等于期望值，就会提示 Mismatch 警告信息。长度匹配是 Mismatch 缺省的匹配方式。

可以使用 lrs_set_receive_option 来设置匹配方式，将缺省值设置为 MISMATCH_CONTENT 来指定匹配方式为内容匹配，那么 lrs_receive 在接收到返回数据后将与期望的数据内容进行匹配对比，这时即使长度相同，如果内容不一样（比如实际接收到的是“A”，而期望数据为“a”），同样会提示 Mismatch 警告信息。

既然提示 Mismatch 警告信息，那么就需要找到这个录制与回放不同的信息，然后对脚本进行关联。进入树模式，这里需要注意一个细节，关联的信息一定是在 Receive Buf 中，因为关联的信息一定是服务器发送给客户端的信息。这里需要关联的信息在 Receive Buf1 中，在 Action 中选择 Receive Buf1，在 Text View（文本视图）中选择要关联的信息后右击，选择 Create Parameter，如图 9-4 所示。

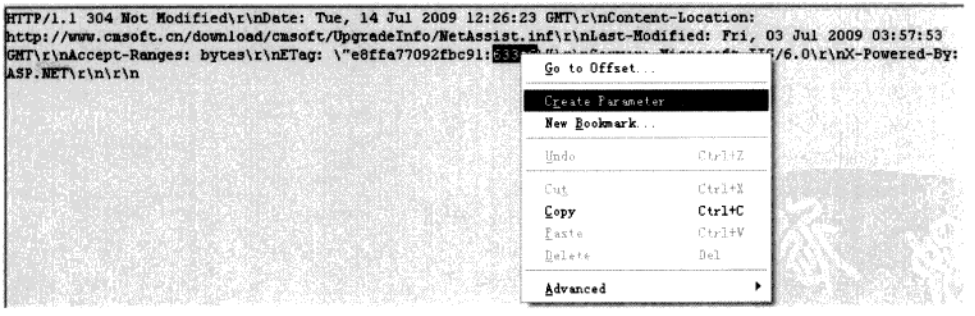


图 9-4 选择要关联的内容

此时，弹出 Create Parameter 对话框，如图 9-5 所示。

“创建参数”对话框中各选项的含义如下：

- Parameter Name（参数名称）：设置定义的参数名称。
- Data Range（数据范围）：设置需要关联信息的偏移量，点击 Select Range 按钮，可以手动更改待关联内容的偏移量。

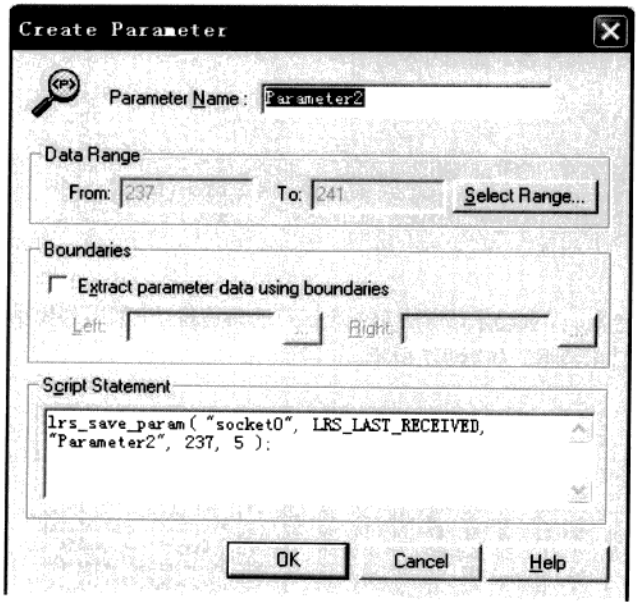


图 9-5 创建参数

- Boundaries（边界）：可以手动设置参数左边界和右边界的值，这样可以固定参数的值。
- Script Statement（脚本声明）：显示生成的关联脚本，即脚本关联完成后脚本中添加的关联函数。当设置了左边界、右边界后发现脚本关联的函数发生了变化。

点击“确定”键，切换回脚本模式，可以看到脚本中多了一行关联的代码。

lrs_save_searched_string("socket0",LRS_LAST_RECEIVED,"port1","LB/BIN=e8ffa77092fbc91:", "RB/BIN=\"\r\\nServer:", 1, 0, -1);

到这里关联工作也已经完成了。

5. 设置运行参数

上面的工作完成之后，脚本编辑已经完成。接下来设置运行时的参数，进入 Vuser→Run-time Settings 对话框，设置运行时的参数即可。

6. 运行脚本

上面的所有工作完成后即运行脚本。

9.1.3 Windows Sockets 数据操作

在使用 Windows Sockets 协议录制后，可以查看并操纵数据。下面介绍几种常用的操纵数据的方法。

1. 查看快照中的数据

在树视图中查看脚本时，VuGen 提供缓冲区快照窗口，可以以文本视图或二进制视图方式查

看快照，并且可以对窗口中显示的数据进行编辑。

文本视图缓冲区快照，以文本形式表示其内容，如图 9-6 所示。

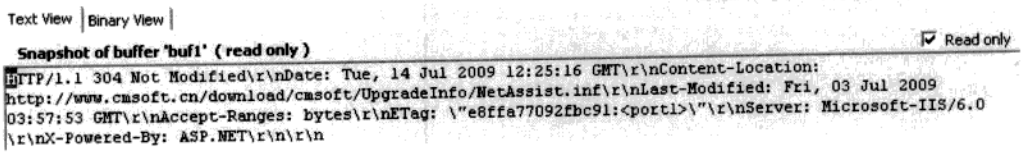


图 9-6 文本视图

二进制视图显示以十六进制表示的数据。左列显示该行中第一个字符的偏移量，中间的列显示数据的十六进制值，右列以 ASCII 格式显示数据，如图 9-7 所示。

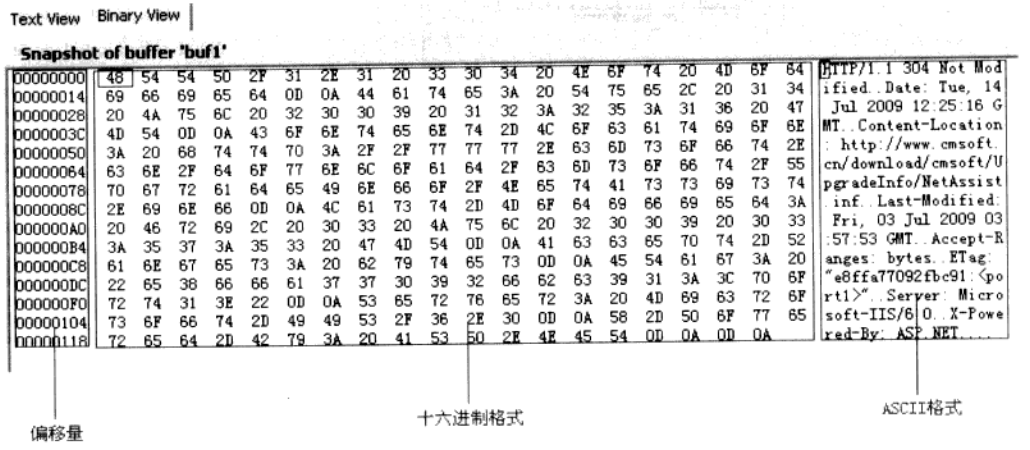


图 9-7 二进制视图

2. 缓冲区导航器

默认情况下，VuGen 左窗格中显示所有的步骤和缓冲区。缓冲区导航器是一个浮动窗口，通过它仅可以显示接收和发送缓冲区（lrs_send、lrs_receive、lrs_receive_ex 和 lrs_length_receive）。此外，还可以应用筛选器，以查看发送或接收缓冲区。

选择 View→Buffer Navigator，打开“缓冲区导航器”对话框，如图 9-8 所示。

当在缓冲导航区选择缓冲区时，其内容会显示在缓冲区快照中。

如果在录制之后更改缓冲区的名称，其内容将不会显示在快照窗口中。要查看已重命名的缓冲区数据，可以使用缓冲区导航器，并选择新缓冲区的名称。VuGen 将发出警告消息，指明选定的缓冲区将禁止创建参数。

3. 转至偏移量

通过指定偏移量，可以在数据缓冲区中移动。可以指定数据的绝对位置，也可以指示与缓冲区

中光标当前位置相对的位置。在快照窗口中点击右键，选择 Go to Offset，将打开 Go to Offset 对话框，如图 9-9 所示。

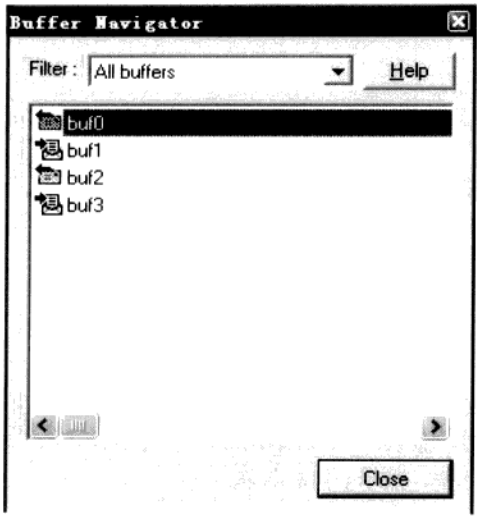


图 9-8 缓冲区导航器

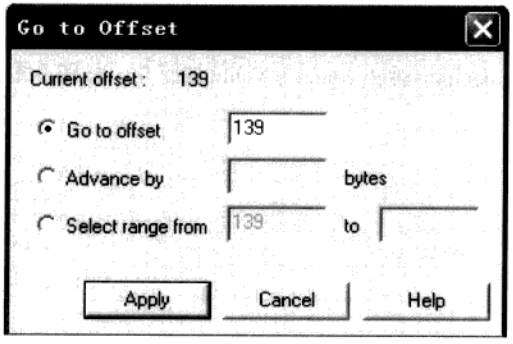


图 9-9 转至偏移量

通过此对话框，可以指定开始和结束偏移量选择数据的范围，并且还可以为关联函数提供要关联数据的偏移量的值。

- Go to offset（转至偏移量）：转至缓冲区中特定的偏移量（绝对偏移量）。
- Advance by（前进）：要跳至与光标相对的位置。输入正值，表示要前进，输入负值，表示在缓冲区内后退。
- Select range from（选择范围）：选择缓冲区中数据的范围，指定开始和结束偏移量。

4. 书签

通过 VuGen，可以将缓冲中的位置标记为书签，并且可以为每个书签赋予一个描述性名称。点击该名称直接跳至该书签的位置。书签列在缓冲区快照下方“输出”窗口的“书签”选项卡中。

在缓冲区快照（文本或二进制视图）中选择一个或多个字节点击右键，选择 New Bookmark，并为书签输入一个描述性名称，如图 9-10 所示。

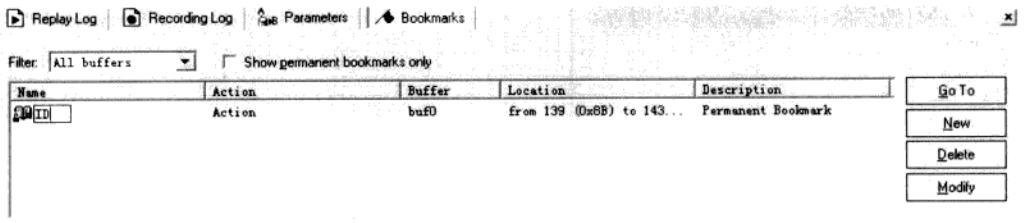


图 9-10 新建书签

书签可以标记单个字节或多个字节。在列表中点击书签时，它在缓冲区快照窗口中将作为选定内容显示。书签的数据内容在文本视图中最初以蓝色突出显示，在二进制视图中，书签则以红色标记。将光标置于缓冲区中的书签上时，将弹出一个文本框，显示书签的名称。

5. 修改缓冲区数据

在树视图中，VuGen 提供了几个工具，使用这些工具，可以通过删除、更改或向现有数据中添加数据对数据进行修改。

数据缓冲区中可以插入数值，可以是单字节、双字节或 4 字节值。

点击右键，选择 Advanced→Insert Number→Specify，弹出 Specify Value 对话框，如图 9-11 所示。

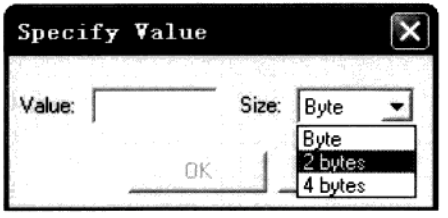


图 9-11 “指定值”对话框

输入想要插入到 Value 框中的 ASCII 值，Size 下拉框中可以选择数据的大小，包括单字节、双字节或 4 字节值，确定后，VuGen 将以十六进制的形式将数据插入缓冲区。

可以对缓冲区数据执行下列所有标准编辑操作：复制、粘贴、剪切、删除和撤消。

执行以上修改缓冲区数据的操作时，一定要保证缓冲区的数据是处于可修改的状态，即 Read only 复选框是未被选中的状态。

9.1.4 关于 LRS 函数

上面对 Windows Sockets 协议录制和使用进行了阐述，下面介绍一些常用的函数。

1. Lrs_accept_connection 接收侦听套接字连接

格式：

int lrs_accept_connection(char *old_socket,char *new_socket);

参数说明：

old_socket：被侦听的套接字标识符，如 socket0。

new_socket：侦听到请求时建立的新套接字标识符，如 socket1。

返回值：

成功返回 0。

例：lrs_accept_connection(socket0,socket1);

函数从 old_socket 上的挂起连接队列中取出第一个连接，使用相同属性创建新的套接字。原来的套接字对于其他连接仍然是可用的。

侦听机制：当 accept 函数监听的 old_socket 收到连接请求时，old_socket 执行体将建立一个新的 socket 连接（标识符为 new_socket），收到服务请求的初始 socket 后（即 old_socket）仍可以继续在以前的 socket 上监听，同时可以在新的 socket 描述符上进行数据传输操作。

2. Lrs_set_receive_option 设置套接字接收选项

格式：

int lrs_set_receive_option (int option,int value,[char *terminator]);

参数说明：

option：接收选项，表示何时停止接收数据。可用选项有 Mismatch 和 EndMarker。

value：选项的值详见表 9-2。

terminator：用于 value 中接收的结束标记。该选项仅在 EndMarker 选项被指定为 StringTerminator 时需要。

表 9-2 option 选项值

option	可用的值
Mismatch	1. MISMATCH_SIZE（缺省值） 2. MISMATCH_CONTENT
EndMarker	1. EndMarker_None（缺省值）——接收全部数据 2. StringTerminator——中断接收的字符串（仅 TCP 连接可用） 3. BinaryStringTerminator——中断接收的二进制标识串（仅 TCP 连接可用） 4. RecordingSize——指定接收长度等于预期长度（仅 TCP 连接可用）

返回值：成功返回 0。

例 1：该例指定对接收到的数据进行内容匹配性验证。

```
lrs_set_receive_option(Mismatch, MISMATCH_CONTENT);
lrs_send("socket1", "buf0", 1, 0);
lrs_receive("socket1", "buf1", LrsLastArg);
lrs_close_socket("socket1");
```

假设预期接收数据中包含一个“%”字符，而回放时该位置接收到的是“#”字符，则该例会报出 Mismatch 信息。如果选项值为 MISMATCH_SIZE 则不会报出 Mismatch 信息。

例 2：该例中 lrs_set_receive_option 函数指定接收的套接字长度等于预期长度。

```
lrs_create_socket("socket0", "TCP", "RemoteHost=199.203.77.12:80", LrsLastArg);
lrs_set_receive_option(EndMarker, RecordingSize);
lrs_receive("socket0", "buf0", LrsLastArg);
```

假如录制时得到的 buf0 的长度为 20 个字节，而回放时从服务器接收到的长度为 30 个字节，以上脚本将只接收前 20 个字节，剩余的 10 个字节将被下一个 lrs_receive 接收。

如果该例的 option 值为 EndMarker_None，那么 lrs_receive 接收完整的返回值并报一个 Mismatch 的消息。

例 3：该例中 lrs_set_receive_option 函数指定接收中断时的二进制标识串

```
lrs_create_socket("socket0", "TCP", "RemoteHost=199.203.77.12:80", LrsLastArg);
lrs_set_receive_option(EndMarker, BinaryStringTerminator, "\\x00\\x07Mercury");
lrs_receive("socket0", "buf0", LrsLastArg);
```

假设服务器发送的内容如下：

```
"\x00\x01\x85\x80\x00\x01\x00\x01\x00\x00\x00\x00\x07"
"Mercury"
"\x02"
"co"
"\x02"
"il"
"\x00\x00\x01\x00\x01\xc0"
```

接收到的内容是：

```
"\x00\x01\x85\x80\x00\x01\x00\x01\x00\x00\x00\x00\x07"
"Mercury"
```

3. lrs_set_send_buffer 指定在套接字上发送的缓冲区

格式：

```
int lrs_set_send_buffer ( char *s_desc,char *buffer,int size );
```

参数说明：

s_desc：套接字标识符。

buffer：指定要发送的缓冲区。

size：发送的字节数。

返回值：发送成功返回 0，失败返回错误码。

```
char *Buffer;
int Size;
```

```
lrs_receive("socket2", "buf20", LrsLastArg);
lrs_get_last_received_buffer("socket2", &Buffer, &Size);
/* 将缓冲区 buf20 中的值保存到用户缓冲区 Buffer 中*/
...
lrs_set_send_buffer("socket2", Buffer, 10 );
lrs_send("socket2","buf21", LrsLastArg );
lrs_free_buffer(Buffer);
```

函数 lrs_set_send_buffer 用来指定下一次调用 lrs_send 时要发送的缓冲区内容。该缓冲区在 lrs_set_send_buffer 中被指定后，其后的一个 lrs_send 中指定的缓冲区内容将不会被发送。

4. lrs_save_param 将静态数据或从缓冲区得到的数据保存到参数中

格式：

int lrs_save_param(char *s_desc,char *buf_desc,char *param_name,int offset,int param_len);

参数说明：

s_desc：套接字标识符。

buf_desc：缓冲区标识符。

param_name：存放缓存数据的参数名称。

offset：被保存到参数中的缓冲区偏移量。

param_len：要保存到参数中的字节数。

返回值：成功返回 0，失败返回错误码。

例 1：保存动态缓冲区数据到参数中。

```
lrs_save_param("socket0","buf0","param1",20,10);
lr_output_message("The content of param1 is%s",lr_eval_string("<param1>"));
```

脚本执行结果：

```
Vuser_init.c(28):lrs_save_param(socket0, buf0, param1,20,3):
Vuser_init.c(29):
The content of param1 is PID
```

例 2：将最后接收到的缓冲区数据保存到参数中。

```
lrs_receive("socket0","buf0",LrsLastArg);
lrs_save_param("socket0",NULL,"param1",20,3);
lr_output_message("the content of param1 is%s:",lr_eval_string("<param1>"));
```

利用该函数将指定的缓冲区数据成功保存到参数中，指定的参数和普通参数一样，可以在脚本的其他地方自由引用。

5. lrs_save_param_ex 将用户、静态或接收到的缓冲区（或缓冲区部分）保存到参数中

格式：

int lrs_save_param_ex(char *s_desc,char *type,char *buff,int offset,int length,char *encoding,char *param);

参数说明：

s_desc：套接字标识符。

type：要将数据保存到参数中的缓冲区类型，有 user（用户缓冲区）、static（data.ws 中的静态缓冲区）和 received（最后接收的缓冲区数据）三种。

buff：和 type 的值有关，如果 type 的值为 user（用户缓冲区），则 buff 的值为指定用户缓冲区；如果 type 的值为 static（data.ws 中的静态缓冲区），则 buff 的值为指定的动态缓冲区；如果 type 的值为 received（最后接收的缓冲区数据），则 buff 的值可设为 NULL。

offset：缓冲区偏移量。

length：保存到参数中的字节数。

encoding：编码方式可以指定为 ASCII 或 EBCDIC，如果是用户缓冲区，则 NULL 默认为 ASCII，如果 type 为 static 或 received，则 NULL 默认为客户端编码方式。

param：参数名称。

返回值：成功返回 0，失败返回错误码。

例 1：保存用户缓冲区数据到参数中。

```
char *userbuffer="chuanshi";
lrs_save_param_ex("socket0","user",userbuffer,0,5,"ascii","param1");
lr_output_message("The content of param1 is %s",lr_eval_string("<param1>"));
```

脚本执行结果：

```
Action.c(9): lrs_save_param_ex(socket0, user, buf_p, 0, 8, ascii, param1)
Action.c(11): The content of param1 is chuanshi
```

例 2：保存静态缓冲区数据到参数中。

```
lrs_save_param_ex("socket0","static","buf0",20,3,"ascii","param1");
lr_output_message("The content of param1 is %s",lr_eval_string("<param1>"));
```

脚本执行结果：

```
Action.c(9): lrs_save_param_ex(socket0, static, buf_p, 20, 3, ascii, param1)
Action.c(11): The content of param1 is PID
```

例 3：将最后接收到的缓冲区数据保存到参数中。

```
lrs_receive("socket0","buf3",LrsLastArg);
lrs_save_param_ex("socket0","received",NULL,20,3,"param1");
lr_output_message("The content of param1 is %s",lr_eval_string("<param1>"));
```

脚本执行结果：

```
Action.c(9): lrs_save_param_ex(socket0, received, buf_p, 20, 3, null, param1)
Action.c(11): The content of param1 is PID
```


函数 lrs_save_param_ex 不识别用户缓冲区的参数化，如例 1 中如果 userbuffer 的内容为“chuan<shi>”（其中<shi>为参数名，值为 hello），那么执行结果将是：Action.c(11): The content of param1 is chuan<shi>

6. lrs_save_searched_string 在静态或接收到的缓冲区中搜索出现的字符串，将出现的字符串的缓冲区部分保存到参数中

格式：

int lrs_save_searched_string (char* s_desc,char* buf_desc,char* param_name,char* left_boundary, char* right_boundary,int ordinal, int offset,int param_len);

参数说明：

s_desc：套接字标识符。

buf_desc：缓冲区标识符。

param_name：参数名称。

left_boundary：标识搜索缓冲区部分左边界的字符串，格式为“LB=xxx”。

right_boundary：标识搜索缓冲区部分右边界的字符串，格式为“RB=xxx”。

ordinal：表示从第几次出现的左边界字符串开始搜索，如果指定了左边界则 ordinal 的值一定大于 0，如果没有指定左边界则将 ordinal 设为-1。

offset：要开始搜索的偏移量。如果指定了左边界则此偏移量相对于左边界计算，否则就从缓冲区的开始处计算偏移量。

param_len：要保存到参数中的缓冲区数据字节数，适用于没有指定右边界的情况。如果指定了右边界则将该参数设置为-1。

返回值：成功返回 0，失败返回错误码。

例 1：将左边界和右边界的值设为 NULL，指定偏移量和字节数。

```
lrs_save_searched_string("socket0","buf0","param1",NULL,NULL,-1,139,5);
lr_output_message("The content is %s",lr_eval_string("<param1>"));
```

数据文件 data.ws 中的内容如下：

```
send buf0 201
"GET /download/cmsoft/UpgradeInfo/NetAssist.inf HTTP/1.1\r\n"
"If-Modified-Since: Fri, 03 Jul 2009 03:57:53 GMT\r\n"
"If-None-Match: \"e8ffa77092fbc91:633a6\"\r\n"
"User-Agent: Internet+Explorer\r\n"
"Host: www.cmsoft.cn\r\n"
"\r\n"
```

脚本执行结果：

```
Action.c(27): lrs_save_searched_string(socket0, buf0, param1, null, null, -1, 139, 5)
Action.c(27): Notify: Saving Parameter "param1 = 633a6"
Action.c(29): Notify: Parameter Substitution: parameter "param1" =   "633a6"
```

例 2：指定左边界值为 e8ffa77092fbc91:，右边界值为\"\r\\nServer，偏移量为 0。

```
lrs_save_searched_string("socket0",LRS_LAST_RECEIVED,"Param1","LB/BIN= e8ffa77092fbc91:", "RB/BIN=\"\r\\nServer",
1, 0, -1);
lr_output_message("The text is:%s", lr_eval_string("<Param1>"));
```

脚本执行结果：

```
Action.c(27): lrs_save_searched_string(socket0, LRS_LAST_RECEIVED, param1, e8ffa77092fbc91:, \"\r\\nServer, 1, 0, -1)
Action.c(27): Notify: Saving Parameter "param1 = 633a6"
Action.c(29): Notify: Parameter Substitution: parameter "param1" =   "633a6"
```

函数 lrs_save_searched_string 将缓冲区中的一部分数据保存到参数中。如果左右边界指定的是二进制字符串，则使用“LB/BIN”或“RB/BIN”来指定，如“LB/BIN=\\x56”。缓冲区标识符表示结构从那个缓冲区中获取数据，并保存到参数中，如果是从数据文件 data.ws 中读取则指定为“bufxxx”，如果设置为 NULL 则表示从最后一个接收到的缓冲区（LRS_LAST_RECEIVED）中读取数据。

该函数的参数列表有以下几种常见的组合方式：

- 左右边界均为 NULL，并指定偏移量和长度（等同于 lrs_sava_param）。
- 指定左边界和右边界出现的次数。
- 指定左边界、左边界出现的次数及读取的数据长度（字节数）。
- 指定左边界、左边界出现的次数、从左边界算起的偏移量及右边界。
- 指定左边界、左边界出现的次数、从左边界算起的偏移量及读取的字节数。
- 只限定偏移量和右边界。

9.2 邮件服务协议

9.2.1 邮件服务协议简介

电子邮件的工作过程遵循客户/服务器模式。每份电子邮件的发送都要涉及到发送方与接收方，发送方构成客户端，而接收方构成服务器，服务器含有众多用户的电子信箱。发送方通过邮件客户程序，将编辑好的电子邮件向邮件服务器（SMTP 服务器）发送。邮件服务器识别接收者的地址，并向管理该地址的邮件服务器（POP3 服务器）发送消息。邮件服务器将消息存放在接收者的电子信箱内，并告知接收者有新邮件到来。接收者通过邮件客户程序连接到服务器后，就会看到服务器的通知，打开自己的电子信箱来查收邮件。

通常 Internet 上的个人用户不能直接接收电子邮件，而是通过申请 ISP 主机的一个电子信箱，由 ISP 主机负责电子邮件的接收。一旦有用户的电子邮件到来，ISP 主机就将邮件移到用户的电子信箱内，并通知用户有新邮件。因此，当发送一封电子邮件给另一个客户时，电子邮件首先从用户计算机发送到 ISP 主机，接着到 Internet，然后到收件人的 ISP 主机，最后到收件人的个人计算机。

ISP 主机起着“邮局”的作用，管理着众多用户的电子信箱。每个用户的电子信箱实际上就是用户申请的账号名。每个用户的电子邮件信箱都要占用 ISP 主机一定容量的硬盘空间，由于这一空间是有限的，因此用户要定期查收和阅读电子信箱中的邮件，以便腾出空间来接收新的邮件。

电子邮件在发送与接收过程中要遵循 SMTP、POP3 等协议，这些协议确保电子邮件在各种不同系统之间的传输。而 LoadRunner 提供的 Mailing Services 的协议有四种：IMAP、MAPI、POP3 和 SMTP。

1. IMAP 协议

IMAP 是 Internet Message Access Protocol 的缩写，顾名思义，主要提供的是通过 Internet 获取

信息的一种协议。IMAP 像 POP 那样提供了方便的邮件下载服务，让用户能进行离线阅读，但 IMAP 能完成的却远远不只这些。IMAP 提供的摘要浏览功能可以让你在阅读完所有的邮件到达时间、主题、发件人、大小等信息后才作出是否下载的决定。

IMAP 是与 POP3 对应的另一种协议，是美国斯坦福大学在 1986 年开始研发的多重邮箱电子邮件系统。它能够从邮件服务器上获取有关 E-mail 的信息或直接收取邮件，具有高性能和可扩展性的优点。IMAP 为很多客户端电子邮件软件所采纳，如 Outlook Express、Netscape Messenger 等，支持 IMAP 的服务器端软件也越来越多，如 CriticalPath、Eudora、iPlanet、Sendmail 等。

IMAP 也是一种用于邮箱访问的协议，使用 IMAP 协议可以在客户端管理服务器上的邮箱，它与 POP 不同，邮件是保留在服务器上而不是下载到本地，在这一点上 IMAP 是与 Webmail 相似的。但 IMAP 有比 Webmail 更好的地方，它比 Webmail 更高效和安全，可以离线阅读等，如果想试试，可以使用 Outlook Express，只要配好一个账号，将邮件接收服务器设置为 IMAP 服务器就可以了。

2. MAPI 协议

MAPI 接口是由微软公司提供的一系列供使用者开发 Mail、Scheduling、Bulletin Board、Communication 程序的编程接口。在使用 MAPI 设计程序时，首先必须在程序和 MAPI 之间建立一个或多个 Session；当 Session 建立好之后，客户端程序就可以使用 MAPI 提供的功能。

MAPI 的功能主要分成三大部分：Address Books、Transport 和 Message Store。Address Books 主要负责设置 E-mail type、protocol 等参数；Transport 负责文件的发送和接收等功能；Message Store 则负责发送接收信息的处理。

3. POP3 协议

POP 的全称是 Post Office Protocol，即邮局协议，用于电子邮件的接收，它使用 TCP 的 110 端口，现在常用的是第三版，所以简称为 POP3。

POP3 仍采用 Client/Server 工作模式。当客户机需要服务时，客户端的软件（Outlook Express 或 FoxMail）将与 POP3 服务器建立 TCP 连接，此后要经过 POP3 协议的三种工作状态，首先是认证过程，确认客户机提供的用户名和密码，在认证通过后便转入处理状态，在此状态下用户可收取自己的邮件或将邮件删除，在完成响应的操作后客户机便发出 quit 命令，此后便进入更新状态，将具有删除标记的邮件从服务器端删除掉。到此为止整个 POP 过程完成。

4. SMTP 协议

SMTP 称为简单邮件传输协议（Simple Mail Transfer Protocal），目标是向用户提供高效、可靠的邮件传输。SMTP 的一个重要特点是它能够在传送中接力传送邮件，即邮件可以通过不同网络上的主机接力式传送。工作在两种情况下：一是电子邮件从客户机传输到服务器；二是从某一个服务器传输到另一个服务器。SMTP 是请求/响应协议，它监听 25 号端口，用于接收用户的邮件请求，并与远端邮件服务器建立 SMTP 连接。

SMTP 通常有两种工作模式：发送 SMTP 和接收 SMTP。具体工作方式为：发送 SMTP 在接到用户的邮件请求后，判断此邮件是否为本地邮件，若是直接投送到用户的邮箱，否则向 DNS 查询远端邮件服务器的 MX 记录，并建立与远端接收 SMTP 之间的一个双向传送通道，此后 SMTP 命

令由发送 SMTP 发出，由接收 SMTP 接收，而应答则反方面传送。一旦传送通道建立，SMTP 发送者发送 MAIL 命令指明邮件发送者。如果 SMTP 接收者可以接收邮件则返回 OK 应答。SMTP 发送者再发出 RCPT 命令确认邮件是否接收到。如果 SMTP 接收者接收到，则返回 OK 应答；如果不能接收到，则发出拒绝接收应答（但不中止整个邮件操作），双方将如此重复多次。当接收者收到全部邮件后会接收到特别的序列，如果接收者成功处理了邮件，则返回 OK 应答。

9.2.2 邮件服务协议录制

这里采用两种常见的协议进行录制：POP3 和 SMTP。这两个协议的区别是：POP3 是负责收邮件的，SMTP 是负责发邮件的。

这里涉及到的软件有：客户端使用 Foxmail 软件，邮件服务器使用 TurboMail 软件。在此进行一个简单的实验，将客户端和服务器安装到同一台机器上，但这并不影响后面的实验。当然实际情况下，服务器和客户端一般都是分开的，具体的网络结构拓扑图如图 9-12 所示。

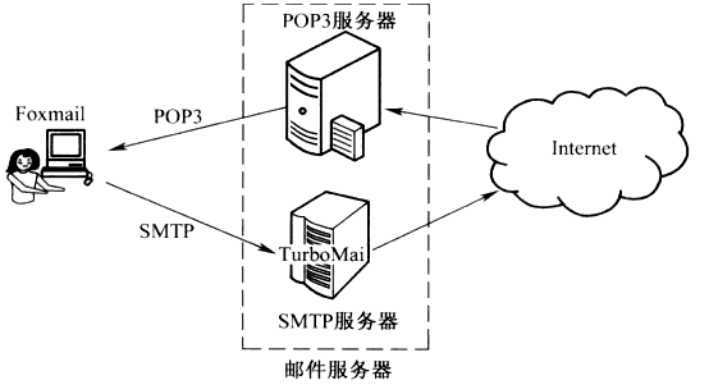


图 9-12 邮件服务器

运行 LoadRunner 的脚本录制器 VuGen，选择 New Multiple Portocol Script 双协议，同时选择 POP3 和 SMTP 两种协议进行录制，如图 9-13 所示。

在录制程序对话框中设置需要录制的程序，这里需要录制的是 Foxmail 程序。这时 LoadRunner 会自动激活 Foxmail 程序。Foxmail 程序运行后，撰写一封邮件并发送。此时发现一个意外的现象，录制过程中 LoadRunner 的那个浮动小窗口的标题栏一直显示“0 events”事件字样，这表明 LoadRunner 并没有捕获到网络通信包。停止录制后，发现里面没有生成相关的脚本语言。

当然这种情况可能并不是每个人都会遇到。在网上也看到很多朋友遇到过类似的情况。如果录制不下来，那么使用 LoadRunner 进行性能测试的希望就变成了泡影。并且没有任何规律，表明什么时候能把脚本录制下来，什么时候录制不下来。一切都是凭运气。

为了解决这种情况下面介绍一种新的录制方法——Port Mapping 录制。

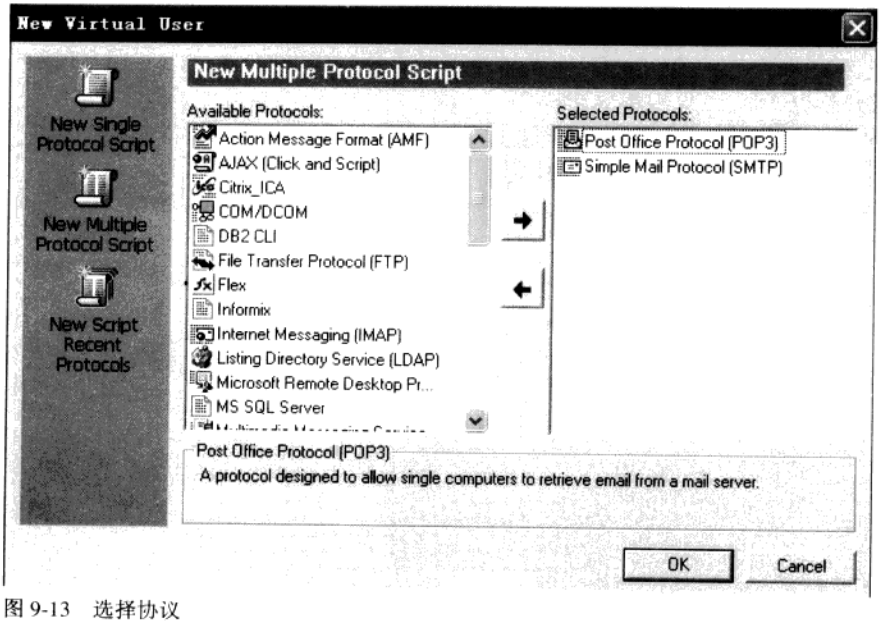


图 9-13 选择协议

选择录制协议的过程还是不变，接着选择录制的程序，在这里不再选择 Foxmail.exe，而是选择 LoadRunner 安装目录下\bin\wplus_init_wsock.exe 程序，如图 9-14 所示。

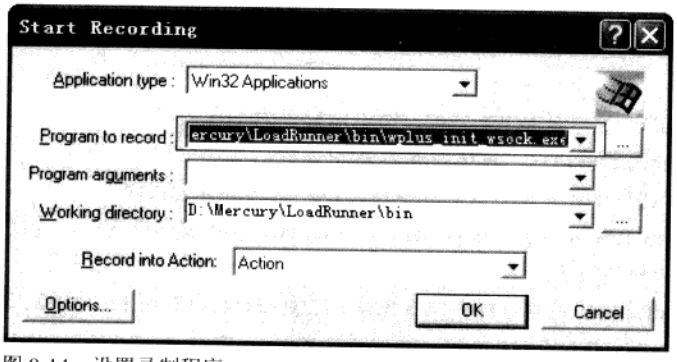


图 9-14 设置录制程序

点击 Options... 按钮，弹出 Recording Options 对话框。选择 Port Mapping 并点击 New Entry 按钮，弹出 Server Entry 对话框。对其进行如图 9-15 所示的设置。

各设置项具体含义如下：

- Target Server（目标服务器）：选择邮件服务器的地址，可以输入 IP 地址或域名地址，如 mail.myspace.com。

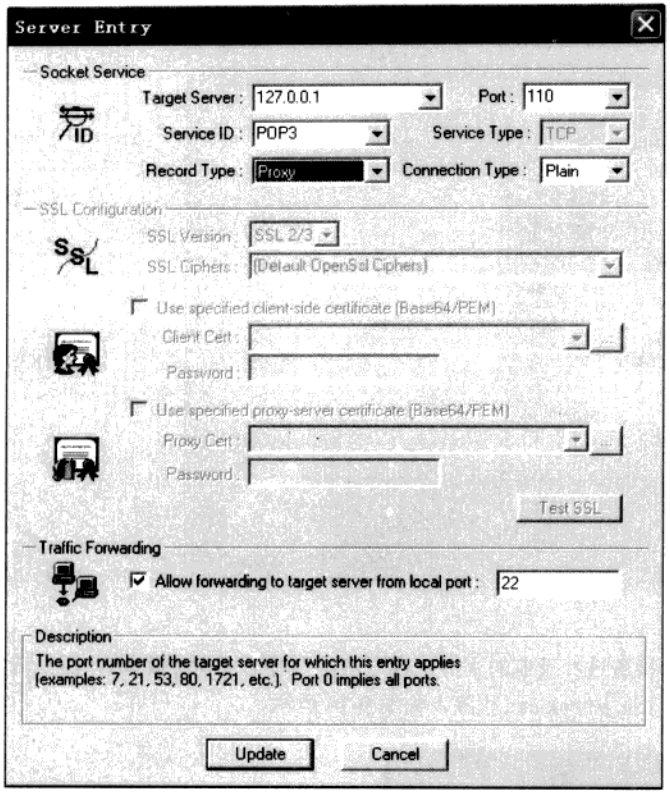


图 9-15 设置代理

- Port（端口号）：设置服务 ID，这里是 POP3 协议，所以端口号为 110。
- Service ID（服务 ID）：选择需要的协议，这里设置为 POP3。
- Record Type（录制方式）：选择录制方式有代理和直接两种，这里设置为代理。
- Traffic Forwarding：将 Allow forwarding to target server from local port 复选框选中，并设置一个端口。这个端口可以随便设置，这里设置为 22。

这些设置项是什么意思呢？LoadRunner 录制过程中，首先得到的是 22 端口的请求，得到 22 端口请求后，将 22 端口请求转发到 127.0.0.1 的 110 端口。其原理如图 9-16 所示。

使用同样的方法，设置好 SMTP 代理。

接下来要对 Foxmail 进行一些设置。进入“邮箱账户设置”对话框，选择“邮件服务器”属性，将发送邮件服务器（SMTP）和 POP3 服务器的地址都设置为 127.0.0.1 或 localhost，如图 9-17 所示。

再在“邮箱账户设置”对话框点击“高级”按钮，进入“高级设置”界面。将 SMTP 服务器端口和 POP3 服务器端口分别设置为 22 和 21。22 和 21 是在设置 LoadRunner 录制选项中 Traffic Forwarding 的端口号，如图 9-18 所示。

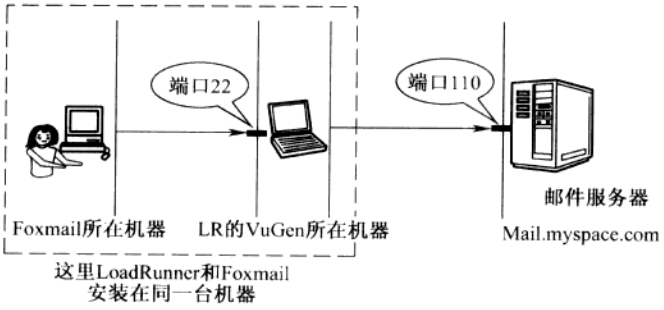


图 9-16 LoadRunner 代理过程

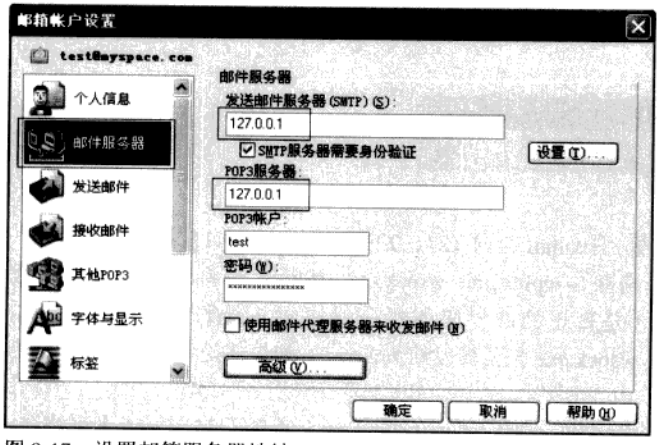


图 9-17 设置邮箱服务器地址

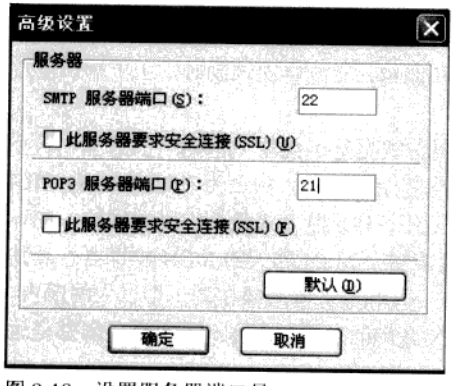


图 9-18 设置服务器端口号

设置完成之后，可以开始录制了，开始录制后，LoadRunner 会激活 wplus_init_wsock.exe 程序，这是一个代理服务器程序，如图 9-19 所示。

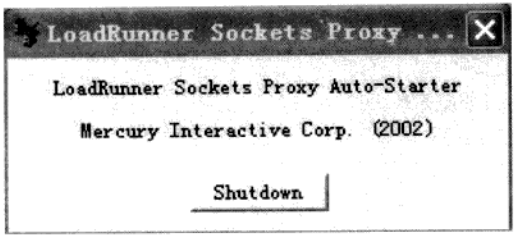


图 9-19 服务器代理程序

这时 LoadRunner 会做一些初始化的工作，会发现在没有做任何动作的情况下 LoadRunner 就已经捕获了网络通信包，如图 9-20 所示。

图 9-20 网络通信包

进行收取一封邮件操作。根据设置，Foxmail 会向 127.0.0.1 即本机 21 端口发送 POP3 命令。这些命令会被 wplus_init_wsock.exe 捕获，wplus_init_wsock.exe 捕获后会将这些命令转发到 mail.myspace.com 的 110 端口。这才是真正的邮件服务器，邮件服务器把信取出后，发给 wplus_init_wsock.exe，然后 wplus_init_wsock.exe 把信转发给 Foxmail。结果就是 Foxmail 里面正确地收取到了邮件。

此时 LoadRunner 显示捕获了 35 个网络包。扣除上面 19 个，实际上捕获了 35-19=16 个网络包。这些网络包就是 Foxmail 发出的取信请求和邮件服务器传递给 Foxmail 的邮件，只是这些请求是经过 wplus_init_wsock.exe 这个代理程序进行转发的。wplus_init_wsock.exe 转发的同时，会把这些通信包记录下来，并转化成脚本，这就是 LoadRunner 录制脚本的真正秘密。

录制结束后，应该把代理程序 wplus_init_wsock.exe 关闭掉，否则再次录制时，由于端口被占用，导致录制无法进行。

通过这个录制过程，可以看出 LoadRunner 录制脚本的方式是基于代理（proxy）的方式。只不过在典型的录制方式中，proxy 是看不见的，或者说不用处理录制方式，称之为“隐式 proxy 录制方式”。而这里介绍的方式称之为“显式 proxy 录制方式”。另外一种录制方式是基于以太网的 Sniffer 方式，利用以太网的通信特性，采用侦听的方式。网上偷窥别人 MSN 聊天的 MSN 嗅探器就是基于这个原理。但是这种 Sinffer 方式的最大缺陷是录制的客户端和服务器必须在同一个以太网段内。Proxy 方式则不受这个限制，甚至可以录制 UNIX 与 UNIX 之间的通信，即客户端和服务器都是运行 UNIX 系统下的录制工作。因此 Proxy 更加具有通用性。

录制完成后生成如下脚本，录制的业务是使用 Foxmail 发送一封邮件，邮件的接收方为自己，

并使用 Foxmail 收取邮件。

```
Action()
{
        smtp1 = 0;
        smtp_logon_ex(&smtp1, "SmtpLogon",
                "URL=smtp://test:1234567@127.0.0.1",
                "CommonName=LoadRunner User",
                LAST);
        smtp_send_mail_ex(&smtp1, "SendMail",
                "To=test@myspace.com",
                "From=<test@myspace.com> SIZE=1443",
                "Subject=Hello",
                "ContentType=multipart/alternative;",
                MAILOPTIONS,
                        "From: \"test\" <test@myspace.com>",
                        "To: \"test\" <test@myspace.com>",
                        "X-mailer: Foxmail 6, 15, 201, 22 [cn]",
                MAILDATA,
                        "AttachRawFile=mailnote1_01.dat",
                        "AttachRawFile=mailnote1_02.dat",
                LAST);
        smtp_logout_ex(&smtp1);
        smtp_free_ex(&smtp1);
        pop31 = 0;
        pop3_logon_ex(&pop31, "Pop3Logon",
                "URL=pop3://test:1234567@127.0.0.1",
                LAST);
        pop3_command_ex(&pop31, "Pop3Command",
                "Command=STAT",
                LAST);
        pop3_list_ex(&pop31, "Pop3List",
                LAST);
        pop3_retrieve_ex(&pop31, "RetrieveMail",
                "RetrieveList=1",
                "DeleteMail=No",
                LAST);
        pop3_logoff_ex(&pop31);
        pop3_free_ex(&pop31);
        return 0;
}
```

9.2.3 脚本分析

录制结束后，可以对脚本进行进一步的分析。

smtp_logon_ex： SMTP Vuser 函数，登录到 SMTP 服务器。

smtp_send_mail_ex： 是 SMTP Vuser 提供的一个最主要的函数，其作用主要是发送 SMTP 消息到指定的 E-mail 地址。

"From: \"test\" <test@myspace.com>"设置发件人地址。

"To: \"test\" <test@myspace.com>"设置收件人地址。

"Subject=Hello"是邮件的主题。

MAILDATA： 发送邮件的内容。

“AttachRawFile=mailnote1_01.dat”和“AttachRawFile=mailnote1_02.dat”是邮件的正文。在脚本左侧导航栏中多了“mailnote1_01.dat”和“mailnote1_02.dat”两个节点。这两个节点中包含的内容如下：

mailnote1_01.dat 的内容如下：

```
Content-Type: text/plain;
        charset="us-ascii"
Content-Transfer-Encoding: 7bit
How are you?
```

这也是发送邮件的正文。

```
mailnote1_02.dat 的内容如下：
Content-Type: text/html;
charset="us-ascii"
Content-Transfer-Encoding: 7bit
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>
<META http-equiv=Content-Type content="text/html; charset=us-ascii">
<META content="MSHTML 6.00.2900.5848" name=GENERATOR><LINK
href="BLOCKQUOTE{margin-Top: 0px; margin-Bottom: 0px; margin-Left: 2em}"
rel=stylesheet></HEAD>
<BODY style="FONT-SIZE: 10pt; MARGIN: 10px; FONT-FAMILY: verdana">
<DIV><FONT face=Verdana size=2>How are you?</FONT></DIV>
<DIV><FONT face=Verdana size=2></FONT> </DIV>
<DIV align=left><FONT face=Verdana color=#c0c0c0 size=2>2009-08-03
</FONT></DIV><FONT face=Verdana size=2>
<HR style="WIDTH: 122px; HEIGHT: 2px" align=left SIZE=2>
<DIV><FONT face=Verdana color=#c0c0c0 size=2><SPAN>test</SPAN>
</FONT></DIV></FONT></BODY></HTML>
```

这段代码和网页的 DOM 源码非常像，类似于 HTML 格式。其实就是发送邮件内容的 text/html 源代码程序。

接下来要对脚本进行参数化，这里要实现的业务模型是模拟不同的用户向 test@myspace.com 邮箱发送邮件。这样涉及到参数化就有两部分，smtp_logon_ex 函数需要进行参数化，使用不同的用户登录 SMTP 服务器；smtp_send_mail_ex 函数中发件箱的地址也需要进行参数化。

参数化后的代码有如下变化：

```
Action()
{
    smtp1 = 0;
    smtp_logon_ex(&smtp1, "SmtpLogon",
        "URL=smtp://{user}:{pw}@127.0.0.1",
        "CommonName=LoadRunner User",
        LAST);
    smtp_send_mail_ex(&smtp1, "SendMail",
        "To=test@myspace.com",
        "From=<{user}@myspace.com> SIZE=1443",
        "Subject=Hello",
        "ContentType=multipart/alternative;",
        MAILOPTIONS,
            "From: \"test\" <test@myspace.com>",
            "To: \"test\" <test@myspace.com>",
            "X-mailer: Foxmail 6, 15, 201, 22 [cn]",
        MAILDATA,
            "AttachRawFile=mailnote1_01.dat",
            "AttachRawFile=mailnote1_02.dat",
        LAST);
    smtp_logout_ex(&smtp1);
    smtp_free_ex(&smtp1);
    pop31 = 0;
    pop3_logon_ex(&pop31, "Pop3Logon",
        "URL=pop3://test:1234567@127.0.0.1",
        LAST);
    pop3_command_ex(&pop31, "Pop3Command",
        "Command=STAT",
        LAST);
    pop3_list_ex(&pop31, "Pop3List",
        LAST);
    pop3_retrieve_ex(&pop31, "RetrieveMail",
        "RetrieveList=1",
        "DeleteMail=No",
        LAST);
    pop3_logoff_ex(&pop31);
```

```
    pop3_free_ex(&pop31);
    return 0;
}
```

参数化结束后，对脚本进行调试。

9.2.4 关于 SMTP 和 POP3 函数

1. smtp_logon_ex 登录到 SMTP 服务器

格式：

int smtp_logon_ex (SMTP *ppsmtp, char *transaction, char *url, [char *CommonName, char *LogonUser, char *LogonPass, char* LocalAddr, char* STARTTLS,] LAST);

参数说明：

ppsmtp：session 标识符。

transaction：定义该步骤的事务名称，事务名使用引号引用；如果不创建事务名，使用空字符（如""）。

url：登录 SMTP 服务器的 URL，如 URL=smtp://user0001t@myspace.com。

commonName：登录 session 的名称，使用“CommonName=***”的格式。

logonUser：可选项，登录 SMTP 服务器的用户名，使用格式“LogonUser=***”。

logonPass：可选项，登录 SMTP 服务器的密码，使用格式“LogonPass=***”。

localAddr：可选项，为客户端设置一个 IP 地址，用于 IP 欺骗，格式为“LocalAddr=192.168.14.52”。

返回值：成功返回 LR_PASS，失败返回 LR_FAIL。

```
smtp_logon_ex(&smtp1, "SmtpLogon",
    "URL=smtp://test:123456@127.0.0.1",
    "CommonName=LoadRunner User",
    LAST);
```

2. smtp_send_mail_ex 发送 SMTP 邮件消息

格式：

int smtp_send_mail_ex (SMTP *ppsmtp, char *transaction, char *RecipientTo, [char *RecipientCC,] [char *RecipientBCC,] char *Subject,[char *From,] [char *ContentType, <char *charset,>] char *MAILOPTIONS, char *MAILDATA, LAST);

参数说明：

ppsmtp：session 标识符。

transaction：定义该步骤的事务名称，事务名使用引号引用；如果不创建事务名，使用空字符（如""）。

recipientTo：邮件接收地址，格式“To=email_address1”。

recipientCC：抄送接收者地址，格式“CC=email_address1”。

recipientBCC：秘密接收者地址，格式“BCC=email_address1”。

subject：邮件主题。

contentType：内容的形式，默认为 multipart/mixed。

MAILOPTIONS：邮件客户端选项与设置，格式为“MAILOPTIONS,"Option1:value", "Option2:value"”。

MAILDATA：邮件的内容，格式为：“MAILDATA, "MessageText=...", "MessageBlob=..."”，当然也可以指定一个附件，格式为："AttachRawFile= filename"，附件的内容一般都是脚本中数据文件。

返回值：成功返回 LR_PASS，失败返回 LR_FAIL。

例：test 向 test001 发送一封邮件，邮件内容为附件 mailnote1_01.dat 和 mailnote1_02.dat 的内容。

```
smtp_send_mail_ex(&smtp1, "SendMail",
        "To=test@myspace.com",
        "From=<test001@myspace.com> SIZE=1443",
        "Subject=Hello",
        "ContentType=multipart/alternative;",
        MAILOPTIONS,
            "From: \"test\" <test@myspace.com>",
            "To: \"test\" <test@myspace.com>",
            "X-mailer: Foxmail 6, 15, 201, 22 [cn]",
        MAILDATA,
            "AttachRawFile=mailnote1_01.dat",
            "AttachRawFile=mailnote1_02.dat",
        LAST);
```

3. Pop_logon_ex 登录 POP3 服务器

格式：

int pop3_logon_ex (POP3 *pppop3, char *transaction, char *url, [char* LocalAddr, char* STARTTLS,] LAST);

参数说明：

pppop3：session 标识符。

transaction：定义该步骤的事务名称，事务名使用引号引用；如果不创建事务名，使用空字符（如""）。

url：定义 POP3 服务器的 URL，格式为"URL=pop3://username:password@server[:port]"。

LocalAddr：可选项，为客户端设置一个 IP 地址，用于 IP 欺骗，格式为“LocalAddr=192.168.14.52”。

返回值：成功返回 LR_PASS，失败返回 LR_FAIL。

例：用户 test 登录 POP3 服务器

```
pop3_logon_ex(&pop31, "Pop3Logon",
        "URL=pop3://test:1234567@127.0.0.1",
        LAST);
```

4. Pop3_retrieve_ex 从 POP3 服务器上获取邮件信息

格式：

long pop3_retrieve_ex(POP3 **pppop3*, char **transaction*, char **retrieveList*,char **deleteFlag*, [<Options>,] LAST);

参数说明：

pppop3：session 标识符。

transaction：定义该步骤的事务名称，事务名使用引号引用；如果不创建事务名，使用空字符（如""）。

retrieveList：设置返回邮件信息列表范围。

deleteFlag：设置当从 POP 服务器上返回邮件后，是否删除邮件的标记。

<Options>：可选项，设置可选参数，可选参数有四个：CreateParamForHeader、ShowMail、SaveTo 和 SaveAs。

返回值：成功返回 LR_PASS，失败返回 LR_FAIL。

```
pop3_retrieve_ex(&pop31, "RetrieveMail",
        "RetrieveList=1",
        "DeleteMail=No",
        LAST);
```

第三部分

实 战 篇

提高篇主要就 LoadRunner 三大组件的使用技巧和高级应用进行了详细的介绍，但并未就 LoadRunner 如何进行性能测试的整个过程做全面的介绍。在实战篇中，将选择两个案例就如何使用 LoadRunner 进行性能测试的整个过程进行全面的介绍。选择的两个实例系统架构模式分别是 B/S 模式和 C/S 模式，目的是通过选取不同架构的系统进行实验。在实验过程中就这两种架构的系统在性能测试过程中的不同之处给予详细说明，并尽可能地在实战过程中引出更多的问题，以期给初学者更多的帮助。

10

客户关系管理系统性能测试

本章选择一个开源的 B/S 架构的系统来讲述如何使用 LoadRunner 做性能测试。该软件是一个开源的软件，故在学习过程中，读者朋友可以下载下来，配置好后，一边学习一边实践。

10.1 系统介绍

本系统是采用最流行的开发语言——PHP 开发语言，基于 LAMP（Linux+Apache+MySQL+PHP）平台开发的新一代 B/S 架构客户关系管理系统。

SugarCRM 是世界领先的、开源的、适合不同规模企业的客户关系管理（CRM）套件，能灵活地应用在各种不同的商业模式上，与一般定制的 CRM 比起来灵活易用且性价比高。SugarCRM 的开源架构让企业能更容易地定制并整合自己的商业流程，从而能建立和维持更好的客户关系。SugarCRM 提供一些开发选项，包括客户化模块、强大的后台实施、在线应用来满足客户的安全、整合、构造需要。

由于该系统是一个开源的系统，故其系统架构不得而知，但其模块关系还是很清楚，主要包括客户关系管理、销售自动化、客户服务跟踪、潜在客户和商机、新闻服务、日历、E-mail、分析报表、统一接口和后台系统管理几大模块，模块关系图如图 10-1 所示。

1. 客户关系管理

为每个客户创建任意数量的联系人。活动历史记录（会议、电话、任务、可带附件的注释和电子邮件）通过联系人、客户、领导、机遇、案例活动可以分配给用户，并支持自动发送邮件功能，以通知用户新任务到达。

2. 销售自动化

管理约会、重要机会、未完成的任务、销售流程图，同时支持每月日历、联系人快速入口。

3. 客户服务跟踪

案例管理系统允许用户跟踪和解决客户问题，记录每个问题的信息周期，以提高客户满意度，每个案例连接到相应的客户、联系人、注释、相关文件以及电话和会议活动的历史记录。

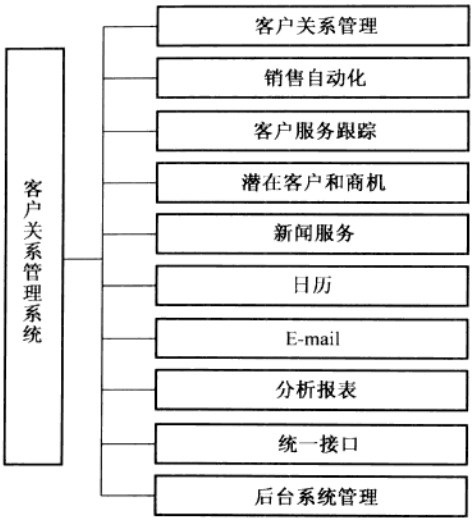


图 10-1 模块关系

4. 潜在客户和商机

用数据分析来找出潜在的商业机会获取更多的客户，实现利益最大化。

5. 新闻服务

RSS 新闻模块允许你将自己喜欢的新闻显示在 My RSS News Feeds 上。

6. 日历

根据日历查看（根据天、星期、月、年）所有的活动，并与相应的任务列表关联起来，共享日历可以查看其他用户的日历以防止日程冲突。

7. 文档

方便的文档管理，便于共享和维护。

8. E-mail

方便的群发邮件，自动提醒，按照分类查找客户邮件。

9. 分析报表

用图形化生动真实地反映目前的销售状况并对将来进行合理的预期分析。

10. 统一接口

门户模块允许管理员和用户将外部的网站和应用程序链接到 SugarCRM 的用户界面，使 SugarCRM 成为用户的一个统一的信息接口。

11. 后台系统管理

用户、角色、权限、个性化修改工作室、邮件营销设置等。

这里要说明的一点是，因为该软件是开源软件，其系统结构架构图不清楚，所以对系统的介绍只能详细到这个程度。

10.2 需求分析

10.2.1 性能指标

性能指标分析，根据客户需求与本系统相结合，用户希望模块能满足以下性能指标，见表 10-1。

表 10-1 用户对系统性能指标的要求

指标种类		用户需求
登录		至少支持一百个左右用户同时并发登录，登录的响应时间不能超过 5 秒
业务	联系人	联系人管理模块包括两个业务过程：进入联系人管理界面和新增联系人并提交。进入联系人管理界面的响应时间不能超过 5 秒，提交新增联系人的响应时间不能超过 8 秒
	客户	客户管理模块包括两个业务过程：进入客户管理界面和新增客户信息。进入客户管理界面的响应时间不能超过 5 秒，新增客户信息的响应时间不能超过 8 秒
	商机	商机管理模块包括两个业务过程：进入商机管理界面和新增商机信息。进入商机管理界面的响应时间不能超过 5 秒，新增商机信息的响应时间不能超过 8 秒
	线索	线索管理模块包括两个业务过程：进入线索管理界面和新增线索。进入线索管理界面的响应时间不能超过 5 秒，新增线索的响应时间不能超过 8 秒

很明显上面的需求是不具可操作性的，这就像和客户谈需求一样，客户只是很简单地描述了需求，而如果仅仅从上面这一个简单的表格来进行性能测试的话，是很难的一件事情，并且很可能测试出来的结果与实际结果存在很大的差距，这样就需要对需求进行详细的分析。

有一点需要说明的是，本书只是借助这个系统来对一个案例的性能测试的实践做深入的分析，故只选择了部分模块进行测试，并没有对整个系统每个模块的性能测试过程进行详细的分析。

10.2.2 需求详细分析

既然上面的需求对实际测试指导意义不强，那么就必须对需求进行进一步的详细分析。

1. 登录

目前情况该公司大概有 700 个员工，但只有 500 个员工使用该系统，部分员工是不使用该系统的，但 5 年后，公司大概有 800 个员工会使用该系统，故确定测试 100 个用户同时并发进行登录操作。所有客户端都在公司的局域网内。

2. 联系人管理模块

联系人管理模块就是导航条中的联系人管理部分，从需求表中很明显可以看出，联系人管理有两部分内容，一是统计进入联系人管理界面的时间，二是新建联系人并提交需求的时间。

并发用户数，将进入联系人管理界面和提交新建联系人信息的用户数设置为相同的用户数。即

使这样这条需求还是有两层意思。一是有多少用户同时在线，二是在线的用户不一定都进行新建联系人活动的操作，也就是说有多少用户在并发进行新建联系人活动。

首先，虽然有 500 个员工会用到这个系统，但统计日常访问量，同时在线的用户大概是在 40 个左右，即使 5 年后也差不多是 60 个人同时使用该系统。这样就确定同时在线的用户大概在 60 个左右。接着，要确定多少用户同时并发进行新建联系人活动，根据日常统计，现在并发用户应该在 15 个左右，即使是 5 年后也大概是在 25 个左右，这样就又确定了同时并发的用户数为 25 个。

但这样还是不够的，现在系统的记录条数比较少，如果 5 年后系统中有大量的联系人记录后情况又将怎么样？根据统计每天新增的联系人记录大概在 10 条左右，一个月大概是 200 条，这样 5 年后大概是 12000 条。

这样就可以很清楚地对它进行性能测试了。

3. 客户管理模块

客户管理模块也包含两部分内容。一是统计进入客户管理界面的时间，二是统计新增一个客户并提交的时间。

这里统计每天同时在线的用户大概在 40 个左右，预计 5 年后大概是 60 个用户同时在线。根据每天的访问量，每天并发访问的用户量不到 15 个，即使是 5 年后测试 25 个用户同时并发即可满足需求。这里进入客户管理界面和新增客户信息都定义为 25 个虚拟用户并发操作。

上面只是定义了并发用户数，但 5 年后客户信息记录条数还不能确定。根据现在的统计每天大概新添 2 条客户记录，一个月大概为 40 条记录，5 年后大概是在 2400 条左右，这样测试就必须在已存在 2400 条记录的情况下进行。

4. 商机管理模块

商机管理模块也包含两部分内容。一是统计进入商机管理界面的时间，二是统计新增商机并提交的时间。

统计每天同时在线的用户大概在 40 个左右，预计 5 年后大概是 60 个用户同时在线。根据每天的访问量，每天并发访问的用户量不到 15 个，即使是 5 年后测试 25 个用户同时并发即可满足需求。进入商机管理界面和新增商机信息都定义为 25 个虚拟用户并发操作。

上面只是定义了并发用户数，但 5 年后商机信息记录条数还不能确定。根据现在的统计每天大概新添 2 条商机记录，一个月大概为 40 条，5 年后也大概是在 2400 条左右，这样测试就必须在已存在 2400 条记录的情况下进行。

5. 线索管理模块

线索管理模块就是导航条中线索管理部分，从需求表中很明显可以看出，线索管理包含两部分内容，一是统计进入线索管理界面的时间，二是新建一个线索并提交需求的时间。

并发用户数方面将进入线索管理界面和提交新线索的用户数设置为相同的用户数。即使这样这条需求还是有两层意思。一是有多少用户同时在线，二是在线的用户不一定都进行销售的操作，也就是说有多少用户在并发进行新增线索操作。

首先，虽然有 500 个员工会用到这个系统，但统计日常访问量的情况，同时在线的用户大概是

在 40 个左右，即使 5 年后差不多也就是 60 个人同时使用该系统。这样就确定同时在线的用户大概在 60 个用户左右。接着，要确定多少用户同时并发进行新增线索活动，根据日常统计可以确定现在并发用户应该在 15 个左右，即使是 5 年后也大概是在 25 个左右，这样就又确定了同时并发的用户数为 25 个。

但这样还是不够的，现在系统的记录条数比较少，如果 5 年后系统中有大量的线索记录后情况又将怎么样？根据统计，每天的线索记录大概在 10 条左右，一个月大概是 200 条，这样 5 年后大概是 12000 条。

这样就可以很清楚地对它进行性能测试了。

到这里需求分析已经完成，根据对这些需求的分析，现在能清楚地知道该如何进行性能测试了。

10.3 测试方案及计划

10.3.1 人力资源

性能测试作为软件测试的一部分工作，并且性能测试一般都是在系统测试完成后，或者是在系统测试阶段中评估系统功能比较稳定，对性能测试没有影响的情况下进行的。根据测试计划，性能测试允许的时间为 25 个工作日，故计划需要 1 个人进行测试。

10.3.2 时间进度

性能测试的计划和时间进度安排，见表 10-2。

表 10-2 性能测试计划

⊟性能测试	25 工作日	2009 年 6 月 8 日	2009 年 7 月 10 日
性能测试用例设计	4 工作日	2009 年 6 月 8 日	2009 年 6 月 11 日
测试环境搭建	3 工作日	2009 年 6 月 12 日	2009 年 6 月 16 日
测试数据准备	3 工作日	2009 年 6 月 17 日	2009 年 6 月 19 日
脚本开发	5 工作日	2009 年 6 月 22 日	2009 年 6 月 26 日
测试执行	5 工作日	2009 年 6 月 29 日	2009 年 7 月 3 日
测试结果分析	5 工作日	2009 年 7 月 6 日	2009 年 7 月 10 日

因为项目的时间周期相对较短，故在此不做里程碑图。

10.3.3 测试环境准备

在进行测试前，必须先搭建好测试平台。

服务器安装操作系统为 Windows 2003 系统，其中数据库服务器和应用服务器安装在同一台机器上，服务器的 IP 地址为 192.168.14.25。

测试机安装的操作系统为 Windows XP 系统，因为测试的并发用户数最多为 100 个，故只要一台测试机即可，其中 Controller 和负载机为同一台机器。测试机与服务器在同一个局域网内。

详细的配置见表 10-3。

表 10-3 测试机与服务器软硬件配置

设备	硬件配置	软件配置
数据库服务器 应用服务器	PC 机（一台） CPU：Intel Xeon X3200 2.4GHz 内存：2.0GB 硬盘：300GB	Windows 2003 MySQL Apache
控制器 负载机	PC 机（一台） CPU：Intel Celeron 3.06GHz 内存：512MB 硬盘：80GB	Windows XP LoadRunner 9.1 IE 6.0 Microsoft Office

测试拓扑结构图，如图 10-2 所示。

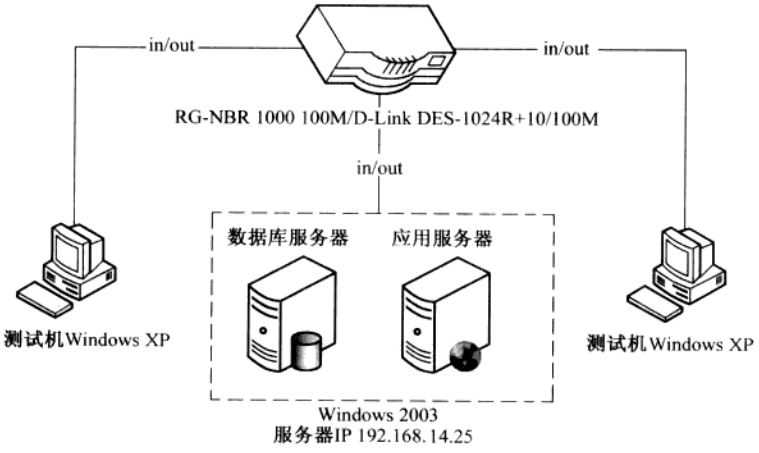


图 10-2 测试拓扑结构图

测试工具：LoadRunner 9.1

录制协议：HTTP/HTML

10.3.4 业务模型创建

测试环境准备好之后要对业务模型进行设计。什么叫业务模型？业务模型是用来约束和规范业务活动的，指导录制脚本时的业务流程及业务背景。如果没有定义好业务模型那么就很难去录制脚本或者是录制好的脚本无法满足客户的需求。这几个模块具体的业务模型，见表 10-4。

表 10-4 业务模型

指标种类		业务模型
登录		100 个虚拟用户同时并发测试
业务	联系人	1. 准备 12000 条联系人记录 2. 进入联系人管理界面的并发用户数为 25 个 3. 新增联系人活动并提交的并发用户数为 25 个
	客户	1. 准备 2400 条客户记录 2. 进入客户界面的并发用户数为 25 个 3. 新增客户记录并提交的并发用户数为 25 个
	商机	1. 准备 2400 条商机记录 2. 进入商机管理界面的并发用户数为 25 个 3. 新增商机记录并提交的并发用户数为 25 个
	线索	1. 准备 12000 条销售活动记录 2. 进入线索管理界面的并发用户数为 25 个 3. 新增线索并提交的并发用户数为 25 个

创建业务模型应该注意以下几点：

1）对于某个业务流程，用户在使用过程中是如何操作的？

2）一个业务包含多个子业务时，子业务的先后顺序和子业务的关系如何处理？

3）为了更好地接近用户的使用习惯，确定业务流程需要哪些支持（如数据准备）。

4）确定虚拟用户并发数和系统在线用户数。

10.3.5 场景模型创建

业务模型是用来规定业务如何活动的，那么场景又如何控制呢？这就需要创建一个场景模型。什么叫场景模型？场景模型用来约束和规范业务活动时的场景环境，指导场景如何设计。也就是说如果没有定义好场景模型那么就无法很好地去定义 Control 部分的场景设计或者测试出来的结果和真实的结果还存在很大的差异。这几个模块具体的场景模型，见表 10-5。

表 10-5 场景模型

指标种类	场景模型
登录	1. 启用脚本中的集合点 2. 每 5 秒加载一个虚拟用户，虚拟用户加载完成后，场景持续运行 5 分钟，结束后，每 5 秒钟释放一个虚拟用户 3. 使用 IP 欺骗，IP 欺骗新建 15 个 IP 地址 4. 添加 Windows 计数器 5. 监控虚拟用户运行日志文件

续表

指标种类		场景模型
业务	联系人	1. 启用脚本中的集合点 2. 每 5 秒加载一个虚拟用户，虚拟用户加载完成后，每 5 秒钟释放一个虚拟用户 3. 使用 IP 欺骗，IP 欺骗新建 15 个 IP 地址 4. 添加 Windows 计数器 5. 监控虚拟用户运行日志文件
	客户	1. 启用脚本中的集合点 2. 每 5 秒加载一个虚拟用户，虚拟用户加载完成后，每 5 秒钟释放一个虚拟用户 3. 使用 IP 欺骗，IP 欺骗新建 15 个 IP 地址 4. 添加 Windows 计数器 5. 监控虚拟用户运行日志文件
	商机	1. 启用脚本中的集合点 2. 每 5 秒加载一个虚拟用户，虚拟用户加载完成后，每 5 秒钟释放一个虚拟用户 3. 使用 IP 欺骗，IP 欺骗新建 15 个 IP 地址 4. 添加 Windows 计数器 5. 监控虚拟用户运行日志文件
	线索	1. 启用脚本中的集合点 2. 每 5 秒加载一个虚拟用户，虚拟用户加载完成后，每 5 秒钟释放一个虚拟用户 3. 使用 IP 欺骗，IP 欺骗新建 15 个 IP 地址 4. 添加 Windows 计数器 5. 监控虚拟用户运行日志文件

创建场景模型应该注意以下几点：

1）确定虚拟用户如何加载？如何释放？以及场景持续运行的时间，这些数据可以通过以往系统使用的历史记录获得，如果以前没有相关的这方面的记录，那么可以通过类似或同行业的情况来做参考。

2）确定集合点使用的情况。

3）确定是否使用 IP 欺骗技术？

4）确定要添加哪些计数器？

10.3.6 测试数据准备

完成以上工作后，接下来就要为业务模型准备数据，一般准备数据可以从以下几个方面入手：

1）数据可以来自于以前的历史数据。如登录模块，测试 10 个用户同时登录的情况，如果已有 10 个真实的用户账号信息，那么在准备数据时，就可以直接调用这些现有的数据。

2）手动添加准备数据。如登录模块，如果现在没有 10 个现成的真实用户账号信息，那么就需要自己手动去创建，当然创建的方式就有很多种了，可以使用 LoadRunner 进行创建，也可以写一

段小程序去创建，当然还可以选择手动创建。但是当数据量很大时，选择手动创建就是一件很困难的事，如测试 BOSS（Business & Operation Support System）系统，几千个虚拟用户并发，如果手动去准备这些数据就很麻烦。

3）数据以何种形式调用。如登录模块的这 10 个用户账号信息，在测试过程中如何调用，这里会出现两种不同的情况。一是文本形式，文本形式有一个缺点是，LoadRunner 参数列表中最多允许 100 行参数，那么如果参数很多就不能用这种方式了，二是数据库的方式，如果大量参数要被调用的话就应选择数据库的形式，因为数据库形式没有受记录的条件限制。

各模块数据准备情况，见表 10-6。

表 10-6 准备数据

指标种类		准备数据
登录		准备好 100 个正确的用户账号信息
业务	联系人	准备好 12000 条联系人记录
	客户	准备好 2400 条客户记录
	商机	准备好 2400 条商机记录
	线索	准备好 12000 条线索记录

这些数据都选择 LoadRunner 生成，100 个用户账号信息存储在数据库中，以方便参数化时调用。

10.4 测试用例

测试用例是进行性能测试过程中最重要的环节之一。一般的，一个性能测试用例，必须包括用例编号、测试目的、并发用户数、模拟用户行为和预期结果这五大部分。

1. 登录

用例编号：LI_001

测试目的：测试 100 个虚拟用户并发时，系统登录的响应时间。

并发用户数：100 个。

模拟用户行为：

1）进入登录界面。

2）输入用户名和密码，点击“登录”按钮。

预期结果：系统登录的响应时间不能超过 5 秒。

2. 进入联系人管理界面

用例编号：TM_001

测试目的：测试进入联系人管理界面活动，系统进入联系人管理界面的响应时间。

并发用户数：25 个。

模拟用户行为：

1）进入登录界面。

2）输入用户名和密码。

3）进入首页，在导航条处点击“联系人管理”按钮，进入联系人管理界面。

预期结果：系统处理进入联系人管理界面的响应时间不能超过 5 秒。

3. 新增联系人

用例编号：TM_002

测试目的：测试提交新增联系人活动，系统提交新增联系人的响应时间。

并发用户数：25 个。

模拟用户行为：

1）进入登录界面。

2）输入用户名和密码。

3）进入首页，在导航条处点击“联系人管理”按钮。

4）在联系人管理界面，点击“新增联系人”按钮。

5）填写待新增联系人信息，并提交。

预期结果：系统处理提交新增联系人信息的响应时间不能超过 8 秒。

4. 进入客户管理界面

用例编号：CL_001

测试目的：测试进入客户界面活动，系统进入客户界面的响应时间。

并发用户数：25 个。

模拟用户行为：

1）进入登录界面。

2）输入用户名和密码。

3）进入首页，在导航条处点击“客户管理”按钮。

预期结果：系统处理进入客户管理界面的响应时间不能超过 5 秒。

5. 新增客户记录

用例编号：CL_002

测试目的：测试提交客户记录，系统提交客户记录的响应时间。

并发用户数：25 个。

模拟用户行为：

1）进入登录界面。

2）输入用户名和密码。

3）进入首页，在导航条处点击“客户管理”按钮。

4）在客户管理界面，点击“新增客户”按钮。

5）填写待新增客户信息，并提交。

预期结果：系统处理提交新增客户信息的响应时间不能超过 8 秒。

6. 进入商机管理界面

用例编号：BC_001

测试目的：测试进入商机管理界面活动，系统进入商机管理界面的响应时间。

并发用户数：25 个。

模拟用户行为：

1）进入登录界面。

2）输入用户名和密码。

3）进入首页，在导航条处点击“商机管理”按钮。

预期结果：系统处理进入商机管理界面的响应时间不能超过 5 秒。

7. 新增商机记录

用例编号：BC_002

测试目的：测试新增商机记录，系统新增商机的响应时间。

并发用户数：25 个。

模拟用户行为：

1）进入登录界面。

2）输入用户名和密码。

3）进入首页，在导航条处点击“商机管理”按钮。

4）在商机管理界面，点击“新增商机”按钮。

5）填写待新增商机信息，并提交。

预期结果：系统处理提交新增商机的响应时间不能超过 8 秒。

8. 进入线索管理界面

用例编号：TH_001

测试目的：测试进入线索管理界面活动，系统进入线索管理界面的响应时间。

并发用户数：25 个。

模拟用户行为：

1）进入登录界面。

2）输入用户名和密码。

3）进入首页，在导航条处点击“线索管理”按钮。

预期结果：系统处理进入线索管理界面的响应时间不能超过 5 秒。

9. 新增线索记录

用例编号：TH_002

测试目的：测试提交新增线索活动，系统新增线索的响应时间。

并发用户数：25 个。

模拟用户行为：

1）进入登录界面。

2）输入用户名和密码。

3）进入首页，在导航条处点击“线索管理”按钮。

4）在线索管理界面，点击“新增线索”按钮。

5）填写待新增线索信息，并提交。

预期结果：系统处理提交新增线索的响应时间不能超过 8 秒。

10.5 执行测试

10.5.1 脚本开发

根据业务模型和场景模型可以开始开发测试脚本，主要涉及到测试脚本实现过程和脚本的结构。虚拟用户脚本的开发情况见表 10-7。

表 10-7 虚拟用户脚本开发情况

用例编号	用例名称	开发情况
LI_001	并发登录	在脚本中对用户名和密码进行参数化，参数调用是通过读取数据库中的数据来获得，设置文本检查点，检查登录的用户名是否正确
TM_001	进入联系人管理界面	该脚本和 LI_001 合并，在 LI_001 登录后，其中有 25 个虚拟用户进行并发进入联系人管理界面操作
TM _002	新增联系人	该脚本和 LI_001 合并，在 LI_001 登录后，其中有 25 个虚拟用户进行并发提交新增联系人活动操作
CL_001	进入客户管理界面	该脚本和 LI_001 合并，在 LI_001 登录后，其中有 25 个虚拟用户进行并发进入客户管理界面操作
CL _002	新增客户记录	该脚本和 LI_001 合并，在 LI_001 登录后，其中有 25 个虚拟用户进行并发新增客户记录操作
BC_001	进入商机管理界面	该脚本和 LI_001 合并，在 LI_001 登录后，其中有 25 个虚拟用户进行并发进入商机管理界面操作
BC _002	新增商机	该脚本和 LI_001 合并，在 LI_001 登录后，其中有 25 个虚拟用户进行并发新增商机记录操作
TH_001	进入线索管理界面	该脚本和 LI_001 合并，在 LI_001 登录后，其中有 25 个虚拟用户进行并发进入线索管理界面操作
TH _002	新增线索记录	该脚本和 LI_001 合并，在 LI_001 登录后，其中有 25 个虚拟用户进行并发新增线索操作

对于脚本的结构分析，在此以登录、进入联系人管理界面和新增联系人三个业务活动为例。其他的大同小异，不做详细介绍。

1. 登录

在录制脚本中定义一个集合点“并发登录”，用来保证虚拟用户真的进行了并发登录操作。定义一个事务“提交登录”，这样来统计登录所花费的时间。添加文本检查点，检查登录的用户名是否正确。所有代码都放在 Action 部分。并发登录用例的脚本结构如图 10-3 所示。

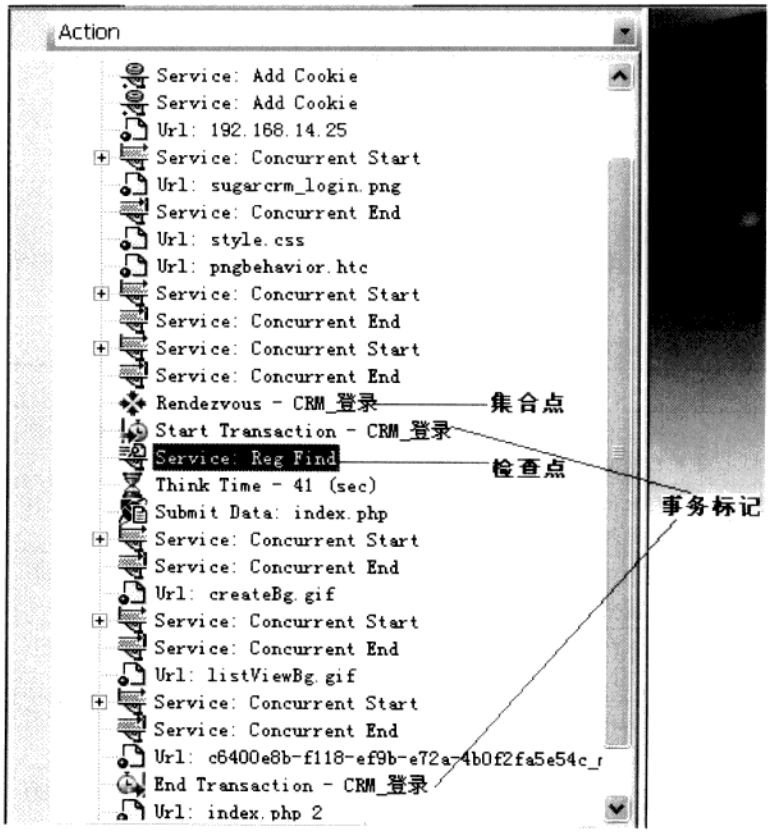


图 10-3 登录用例脚本结构

对登录的用户名进行参数化，将参数化的文件放在一个专门管理参数化数据的文件夹中，将参数列表中读取参数的路径由绝对路径更改为相对路径。在这里并未对密码进行参数化，为了测试的方便，在准备数据时，用户名密码统一设置为“1”。故不需要对密码进行参数化。

2. 进入联系人管理界面

在进入联系人管理界面的脚本中，只需对登录的用户名进行参数化，这里没有对密码进行参数化，因为所有用户名密码都是一样的。设置集合点，确保所有虚拟用户并发进入联系人管理界面。其脚本结构图，如图 10-4 所示。

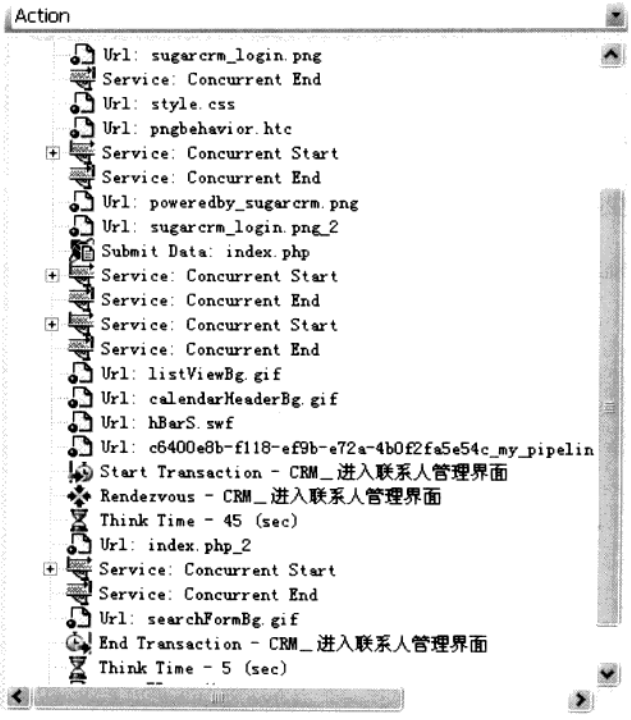


图 10-4 进入联系人管理界面脚本结构

3. 新增联系人信息

新增联系人脚本是最重要的脚本之一。录制完成脚本之后，对脚本进行回放，回放后进入系统查看是否已添加脚本中的新增联系人信息，此时发现，该脚本中新增用户信息并未被添加（录制脚本使用的登录用户名为 test1，添加的联系人姓名为 hehe1，脚本调试过程中对登录用户名进行参数化，参数内容为 test1 和 test2），即用户 test2 中并没有联系人 hehe1 的信息，这说明脚本出现问题。这时分析回放日志，但并未发现异常日志，这时一种猜测是脚本应该需要关联，前面讲了脚本关联的几种方法，下面进行以下分析处理：

1）使用两个用户 test1 和 test2 分别录制业务流程一样的脚本。

2）再使用 LoadRunner 自带的 WDiff 工具比较两个脚本，此时发现在提交新增联系人信息的过程中“Name=assigned_user_id”的内容不一致，如图 10-5 所示。

```
"Name=reports_to_id", "Value=", ENDITEM,
"Name=assistant", "Value=", ENDITEM,
"Name=assistant_phone", "Value=", ENDITEM,
"Name=assigned_user_id", "Value=d6dc0d7f-fb3a-8452-a418-4b0f2cdbd8b6"
"Name=invalid_email", "Value=0", ENDITEM,
"Name=primary_address_street", "Value=", ENDITEM,
```

图 10-5 WDiff 比较脚本

这说明每次提交一个新增联系人信息时，“Name=assigned_user_id”的值是服务器分配的，并且每次的值都不一样，这就说明需要对该 ID 进行关联，接着找到需要关联 ID 的左右边界，手动创建一个关联规则，重新录制脚本即可，关联后代码如图 10-6 所示。

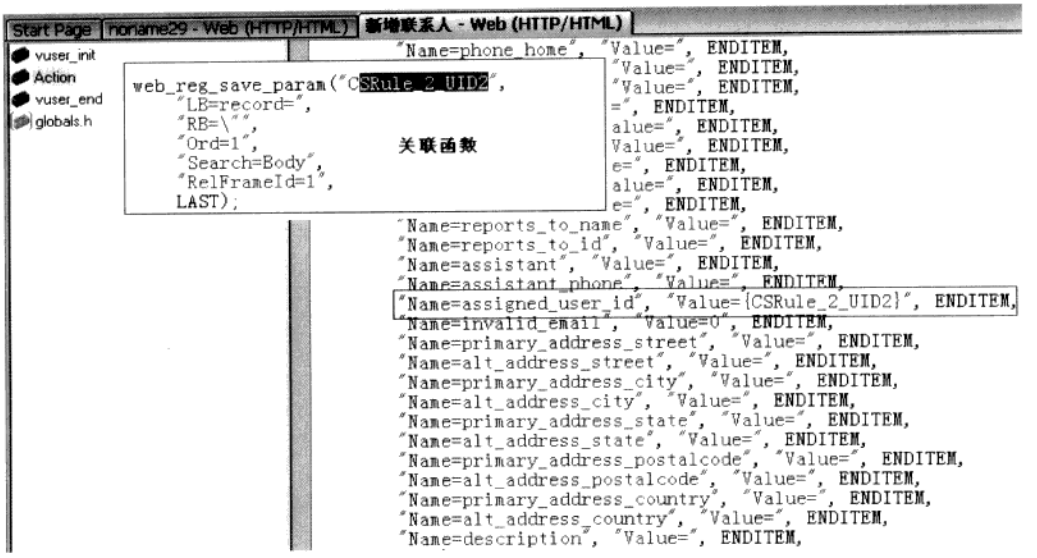


图 10-6 脚本关联

关联后再次进行回放，发现 test2 下面还是没有添加那个联系人，这说明脚本还是有问题，但实在是想不出来哪里有问题，然后手动用 test2 这个用户名登录系统，添加刚才的联系人，结果提示：“存在同名的联系人，是否确定添加”。到这里说明脚本已经成功添加了联系人，只是在联系人列表中未看到添加的联系人信息，如图 10-7 所示。

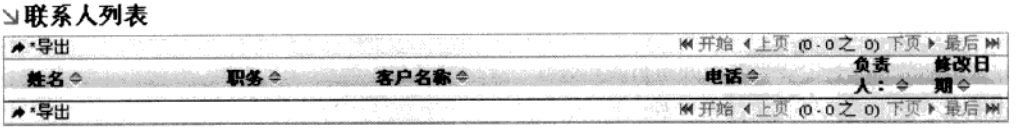


图 10-7 联系人列表

这说明脚本在处理业务流程时已经出现问题，接下来应该分析添加联系人的业务流程，添加联系人界面如图 10-8 所示。

经过分析发现，在提交新增联系人信息时，有一项“负责人”一直都是当前的用户名，再看一下联系人列表中“负责人”栏，只有当前负责人添加的联系人信息才会显示出来，而 test2 用户下其实是添加了 hehe1 这个联系人的，之所以看不到是因为在脚本运行的时候“负责人”项一直都是 test1，原因找到了，最后只需要对“负责人”项进行参数化，并且负责人应该和登录的用

户名保持一致，但问题是不知道脚本中哪一项是负责人信息，又没有其他的技巧，后来经过多次手动试验一项一项的排除后才找到了“负责人”项在脚本的位置，下面是修改后的脚本，如图 10-9 所示。

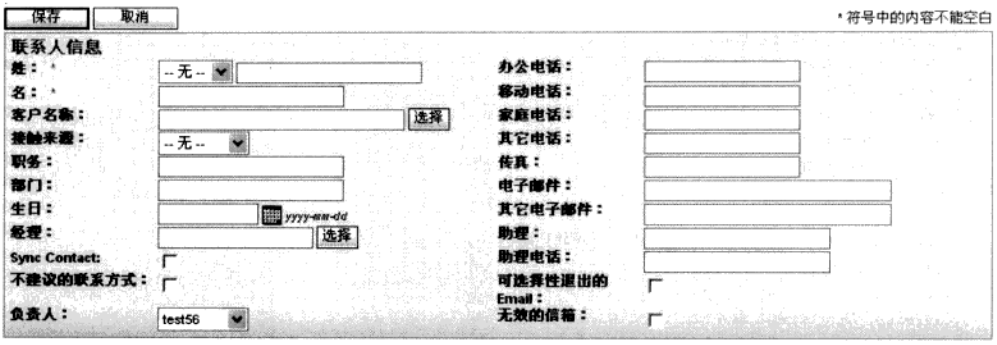


图 10-8 添加联系人

```
web_submit_data("index.php_4",
    "Action=http://192.168.14.25/index.php",
    "Method=POST",
    "RecContentType=text/html",
    "Referer=http://192.168.14.25/index.php?module=Contacts&action=EditView&
    "Snapshot=t73.inf",
    "Mode=HTTP",
    ITEMDATA,
    "Name=module", "Value=Contacts", ENDITEM,
    "Name=record", "Value={user}", ENDITEM,
    "Name=action", "Value=Save", ENDITEM,
    "Name=opportunity_id", "Value=", ENDITEM,
    "Name=contact_role", "Value=", ENDITEM,
    "Name=case_id", "Value=", ENDITEM,
    "Name=bug_id", "Value=", ENDITEM,
    "Name=return_module", "Value=Contacts", ENDITEM,
    "Name=return_id", "Value=", ENDITEM,
    "Name=return_action", "Value=index", ENDITEM,
    "Name=button", "Value=  淇濆瓨  ", ENDITEM,
    "Name=salutation", "Value=", ENDITEM,
    "Name=first_name", "Value=Li", ENDITEM,
    "Name=phone_work", "Value=", ENDITEM,
    "Name=last_name", "Value={lastname}", ENDITEM,
    "Name=phone_mobile", "Value=", ENDITEM,
    "Name=account_name", "Value=", ENDITEM,
    "Name=account_id", "Value=", ENDITEM,
    "Name=phone_home", "Value=", ENDITEM,
```

该项值是负责人的值

图 10-9 对负责人的值进行参数化

再次回放脚本，脚本顺利的完成任务。

新增联系人脚本结构图，如图 10-10 所示。

其他模块脚本开发和进入联系人管理界面、新增联系人信息的脚本大同小异，故在这里就不对剩下的模块进行一一的描述。

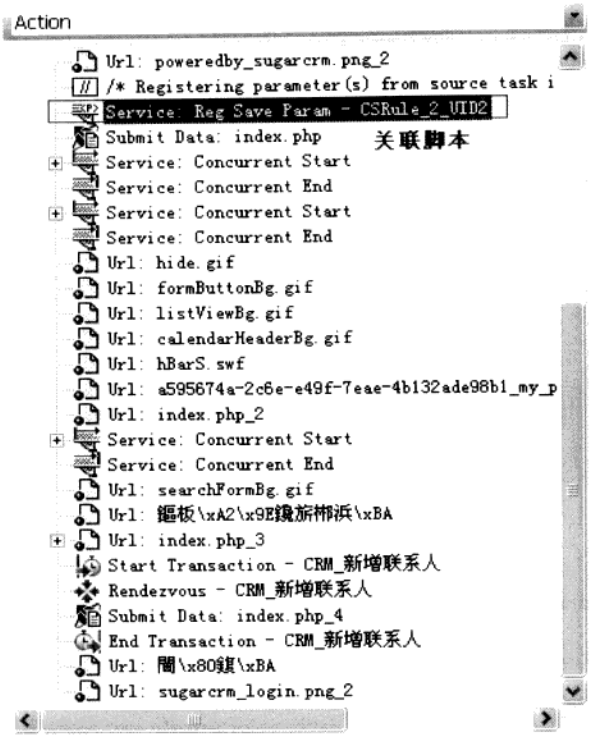


图 10-10 脚本结构图

10.5.2 场景设计

场景设计主要是对 Controller（控制器）进行设置，设置脚本运行时的环境。这里只对登录、进入联系人管理界面和新增联系人三个模块进行详细的描述，其他的模块都大同小异，在此不进行详细的描述。

1. 登录

这里场景组设置 100 个虚拟用户，如图 10-11 所示。

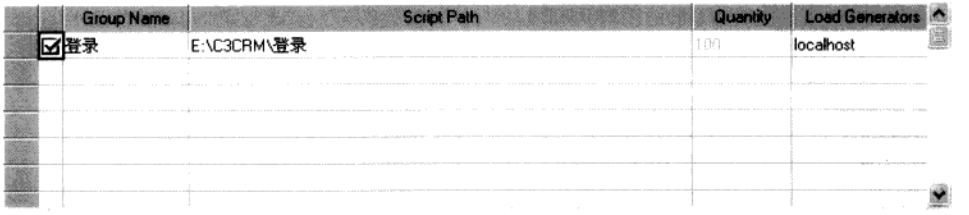


图 10-11 场景组设计

在这里没有对负载发生器（Load Generators）进行设置，因为在试验时只使用了一台机器作负载发生器，并且负载发生器和控制器是在同一台机器上，故看到的负载发生器只有 Localhost，但是如果在测试过程中有多台时，就得对负载发生器进行配置。还有一点就是如果有多台负载发生器，为了达到负载均衡的目的，需要将所有的负载平均地施压到服务器上，即负载均衡技术。

场景策略的设置，在脚本运行时对所有的虚拟用户进行初始化，每 5 秒加载一个虚拟用户，虚拟用户加载完成后，持续运行 5 分钟，运行结束后每 5 秒释放一个虚拟用户，直到所有虚拟用户释放完成，如图 10-12 所示。

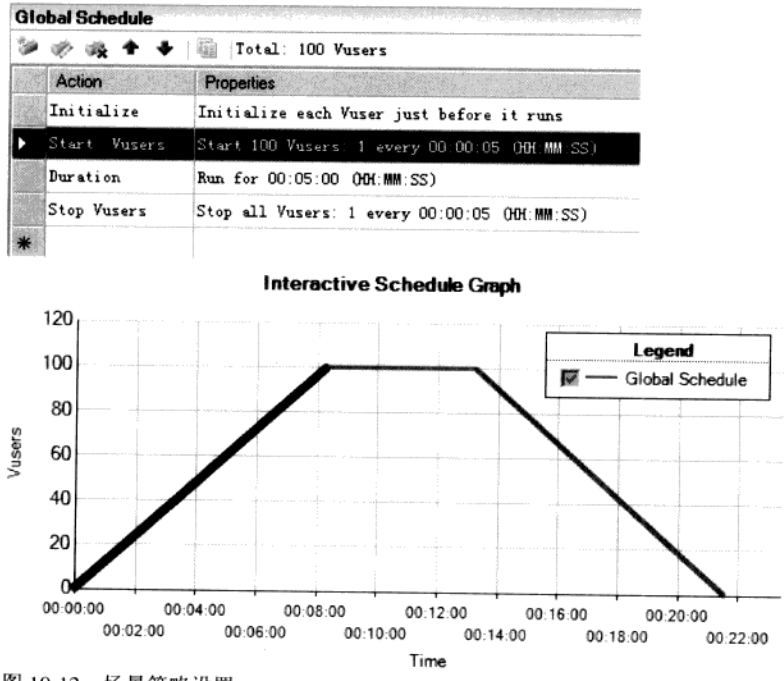


图 10-12　场景策略设置

启动 IP 欺骗功能，脚本中所有集合点都设置为使用的状态，如图 10-13 所示。

2．进入联系人管理界面

场景组设置 25 个虚拟用户，如图 10-14 所示。

和登录模块一样，负载发生器没有进行设置。

场景策略设置，在脚本运行时对所有的虚拟用户进行初始化，每 5 秒加载一个虚拟用户，虚拟用户加载完成后，持续运行 5 分钟，运行结束后每 5 秒释放一个虚拟用户，直到所有虚拟用户释放完成，如图 10-15 所示。

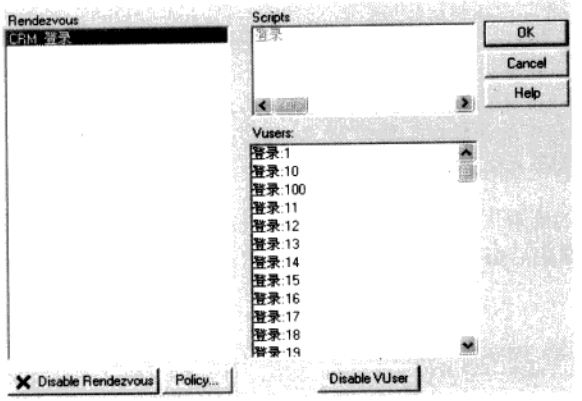


图 10-13 集合点设置

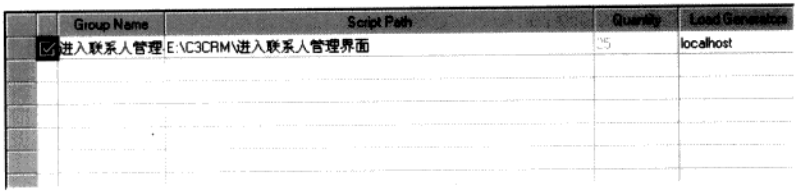


图 10-14 场景组设置

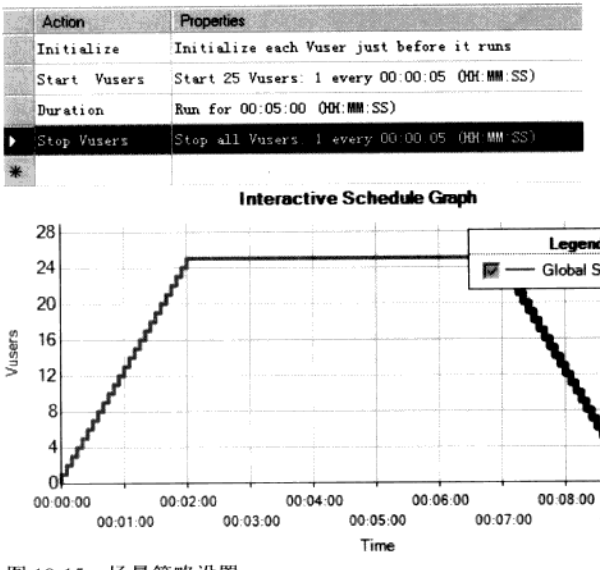


图 10-15 场景策略设置

启动 IP 欺骗功能，脚本中所有集合点都设置为使用的状态，如图 10-16 所示。

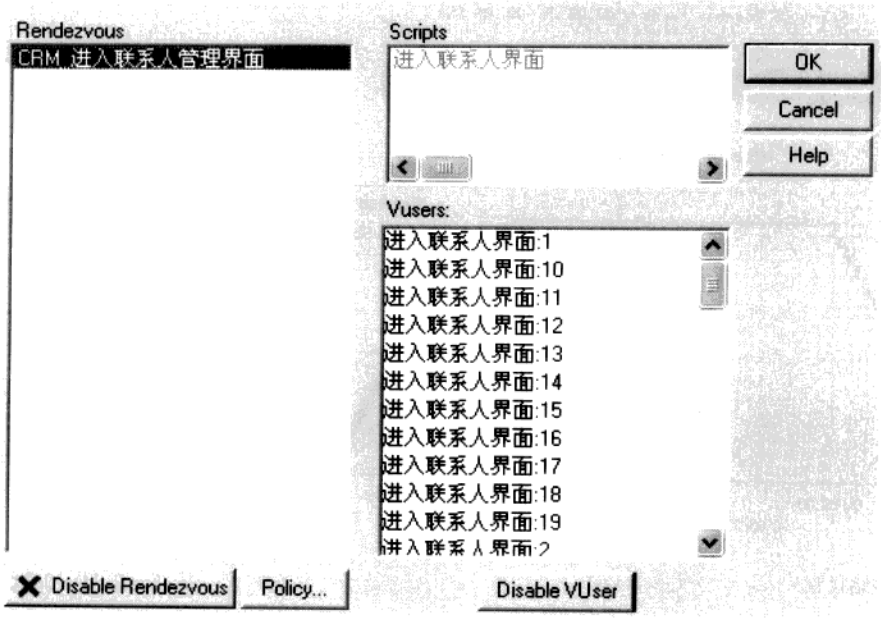


图 10-16 集合点设置

3．新增联系人信息

场景组设置 25 个虚拟用户，如图 10-17 所示。

Group Name	Script Path	Quantity	Load Generators
新增联系人	E:\C3CRM\新增联系人	25	localhost

图 10-17 场景组设置

和登录模块一样，负载发生器没有进行设置。

场景策略设置，在脚本运行时对所有的虚拟用户进行初始化，每 5 秒加载一个虚拟用户，虚拟用户加载完成后，持续运行 5 分钟，运行结束后每 5 秒释放一个虚拟用户，直到所有虚拟用户释放完成，如图 10-18 所示。

启动 IP 欺骗功能，脚本中所有集合点都设置为使用的状态，如图 10-19 所示。

	Action	Properties
	Initialize	Initialize each Vuser just before it runs
	Start Vusers	Start 25 Vusers: 1 every 00:00:05 (HH:MM:SS)
	Duration	Run for 00:05:00 (HH:MM:SS)
▶	Stop Vusers	Stop all Vusers: 1 every 00:00:05 (HH:MM:SS)
*		

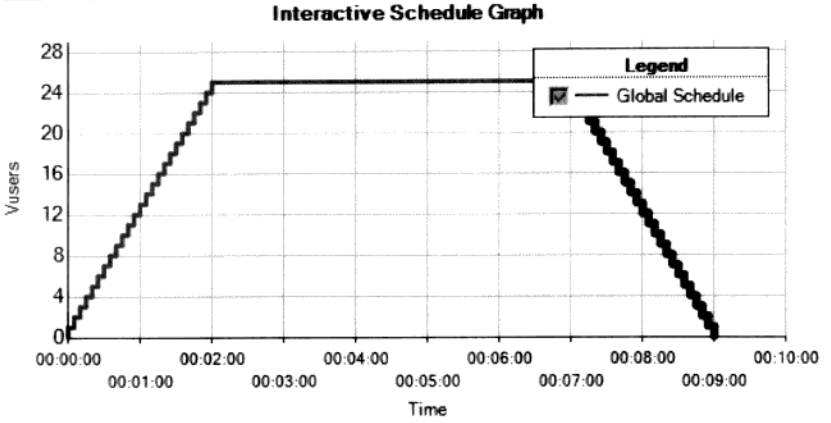


图 10-18 场景策略设置

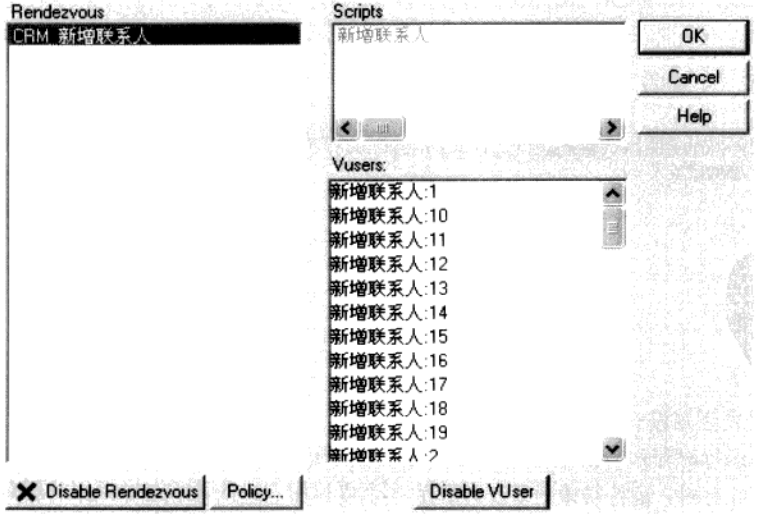


图 10-19 集合点设置

其他的几个模块在这里就不一一介绍了，都是大同小异。

10.5.3 计数器设置

计数器设置主要是设置在场景运行时，需要监控哪些资源。在这里所有的脚本都对 Windows 资源和数据库服务器进行监控。这里以登录模块为例，其他的模块设置一致。

1. Windows 计数器

添加 Windows 计数器，具体如何添加在前面讲述过，这里就不再详细描述，监控的对象为需要监控的某台服务器（数据库服务器或应用服务器的 Windows 资源），当然因为这里两个服务器都安装在同一台机器，不存在这些之分。Windows 资源计数器如图 10-20 所示。

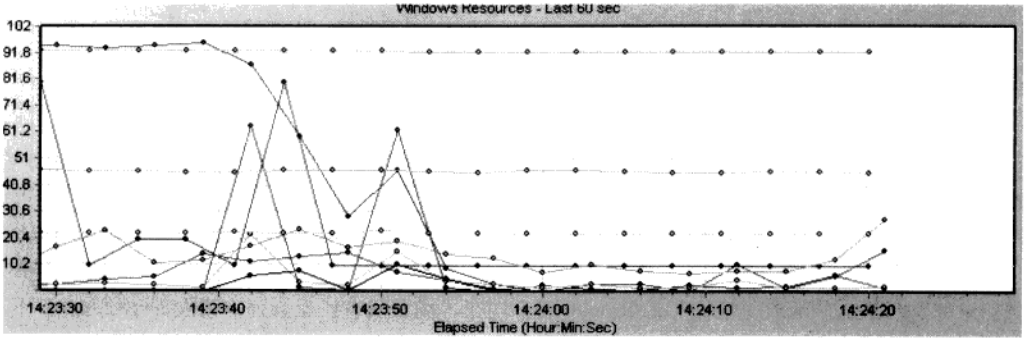


图 10-20 Windows 资源监控图

2. MySQL 数据库计数器

MySQL 数据库计数器的添加方式与 Windows 计数器的添加方式一样。被监控的机器为数据库服务器。MySQL 数据库计数器如图 10-21 所示。

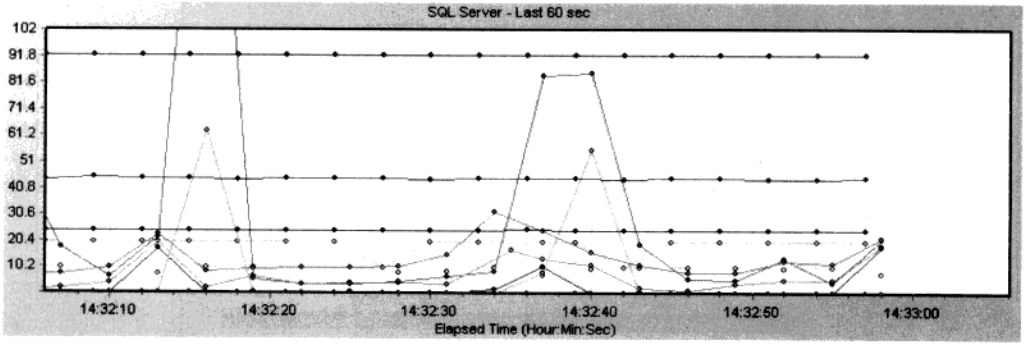


图 10-21 MySQL 数据库计数器

在添加数据库计数器时，有时场景状态栏输出窗口会提示如图 10-22 所示的错误信息。

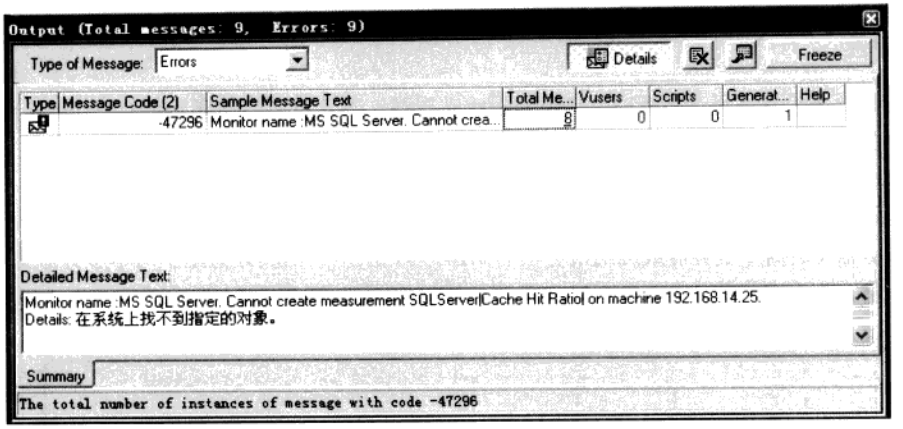


图 10-22 数据库计数器添加报错

出错的原因是计数器的这些信息和 Windows 的这些信息名称不相符引起的，有时候在 Windows 计数器中也会出现。出错后这些指标的值 LoadRunner 无法获取。但是为什么与 Windows 资源的名称不相符就会报错，并且没有值显示呢？原因是因为 LoadRunner 的这些计数器的值其实不是自己创建的，而是从 Windows 系统中调用出来的，故如果名称不一样就无法从 Windows 系统中读到这些指标的值，这样这些指标的值就无法被 LoadRunner 显示出来，场景状态栏就会弹出错误提示信息。

Windows 操作系统内部监视器如图 10-23 所示。

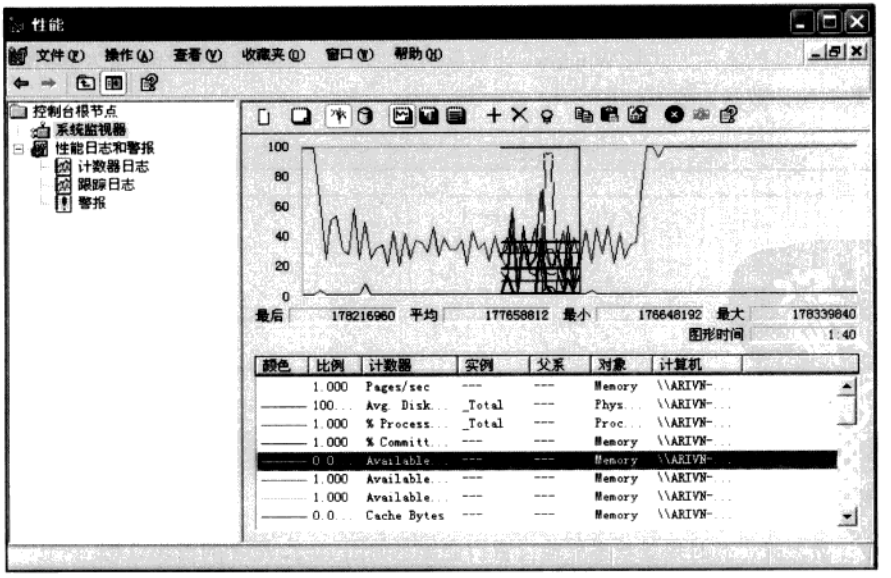


图 10-23 Windows 操作系统监视器

Windows 操作系统监视器也可以添加计数器，这和 LoadRunner 添加的计数器如出一辙。所以其实这些指标并不是 LoadRunner 自己去创建的，而是直接从 Windows 操作系统中“挖过来的”。

10.5.4 场景监视

在场景运行过程中必须对场景进行监控，通过监控场景运行时的情况来获得一些信息，这样有利于对性能测试结果进行分析，以下几个方面的信息需要监控。

1. 场景组运行控制信息

在这里需要监控场景组中所有虚拟用户运行的情况，同时也可以对虚拟用户运行的情况进行控制，如图 10-24 所示。

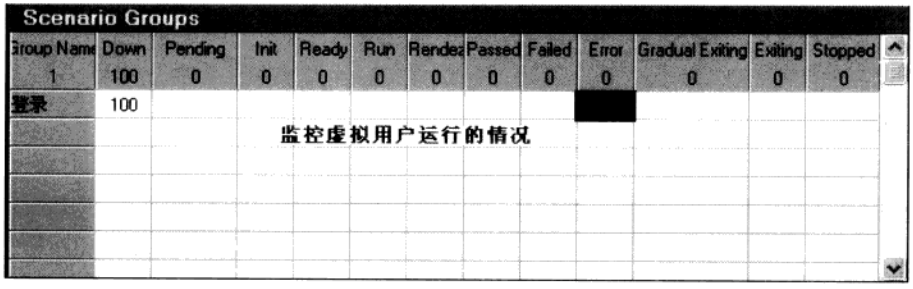


图 10-24 场景组中虚拟用户运行的情况

同时还需要监视虚拟用户组中每个虚拟用户运行的情况，并且一定要观察日志文件的情况，如图 10-25 所示。

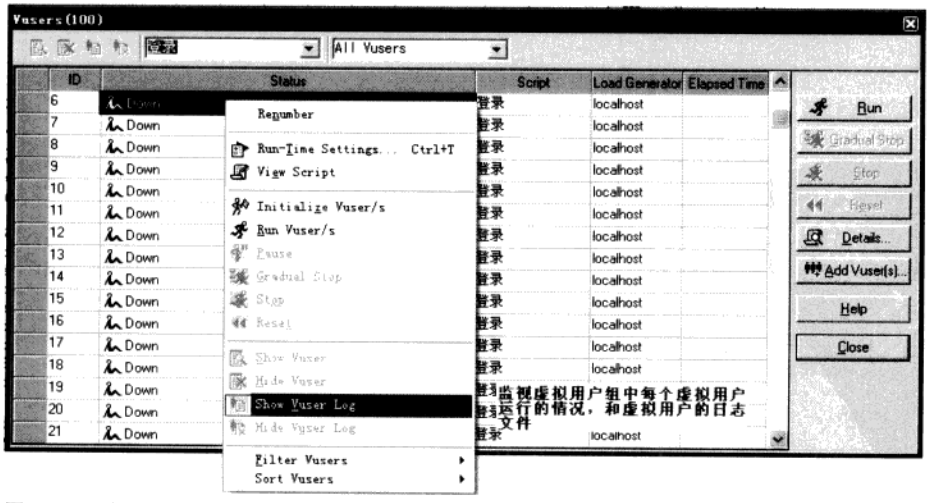


图 10-25 监视虚拟用户组运行的情况和日志文件

2. 监视场景状态图

监视场景状态图中信息的情况，最重要的是关注错误事务的情况，如图 10-26 所示。

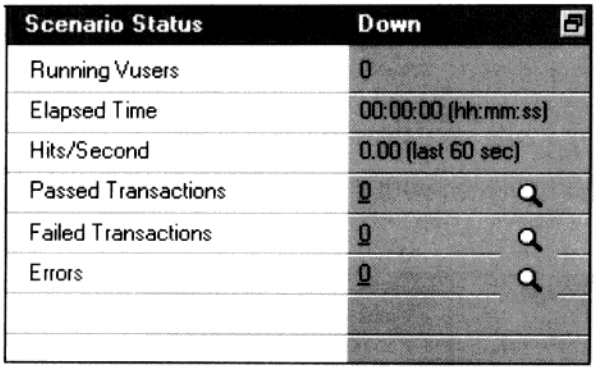


图 10-26 监视场景运行状态图

3. 监视输出对话框

如果在运行过程中场景状态出现 Errors 的信息，那么一定要查看输出对话框错误信息的情况，可以帮助调试脚本和分析结果，如图 10-27 所示。

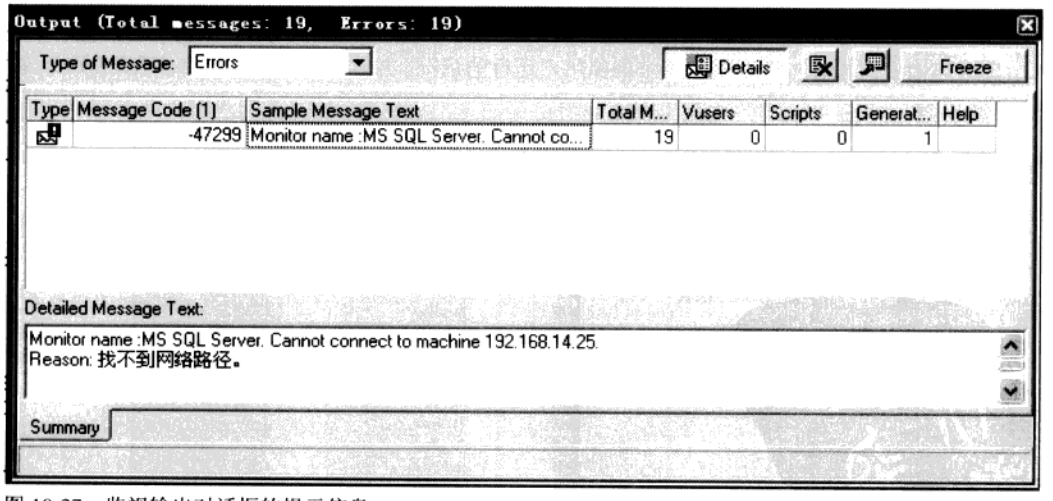


图 10-27 监视输出对话框的提示信息

4. 监视数据图

在数据图部分主要监视 5 个视图的变化（正运行虚拟用户数、事务的响应时间、每秒点击率、Windows 计数器和 MySQL 计数器），如图 10-28 所示。

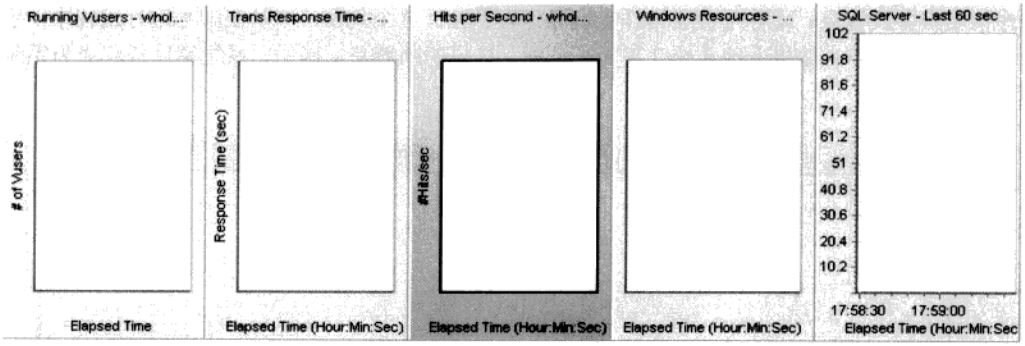


图 10-28 监视数据图的信息

10.6 结果分析

脚本执行完成后，就得分析测试结果，下面是每个模块执行的结果。

1. 登录

场景是模拟 100 个虚拟用户并发登录，当虚拟用户加载到 50 个时，每加载一个虚拟用户，场景状态栏的失败事务数和错误信息就在增多，如图 10-29 所示。这说明当加载到 50 个虚拟用户后，服务器无法处理客户端的请求。

Scenario Status	Down
Running Vusers	0
Elapsed Time	00:27:40 (hh:mm:ss)
Hits/Second	0.00 (last 60 sec)
Passed Transactions	288
Failed Transactions	13980
Errors	27007

图 10-29 失败事务和错误信息

接下来分析平均事务响应时间，如图 10-30 所示。平均事务响应时间一直在增加，也同样说明服务器无法处理客户端的请求，事务一直无法处理完成。到这里可以得出结论应该是服务器已经出现问题，但还不明确是什么原因导致的。

再看一下 Windows 计数器捕捉到的数据，如图 10-31 所示。在这里很明显地看到 CPU 的使用率达到 100%，内存也存在问题，但是网络没有问题，这说明服务器的硬件配置无法处理 100

个虚拟用户并发登录，硬件平台成为性能瓶颈。为了验证这个判断，可以在脚本运行过程中，手动登录一次试一下，结果发现系统几乎无法动弹。这说明判断是正确的，系统硬件资源成为系统性能的瓶颈。

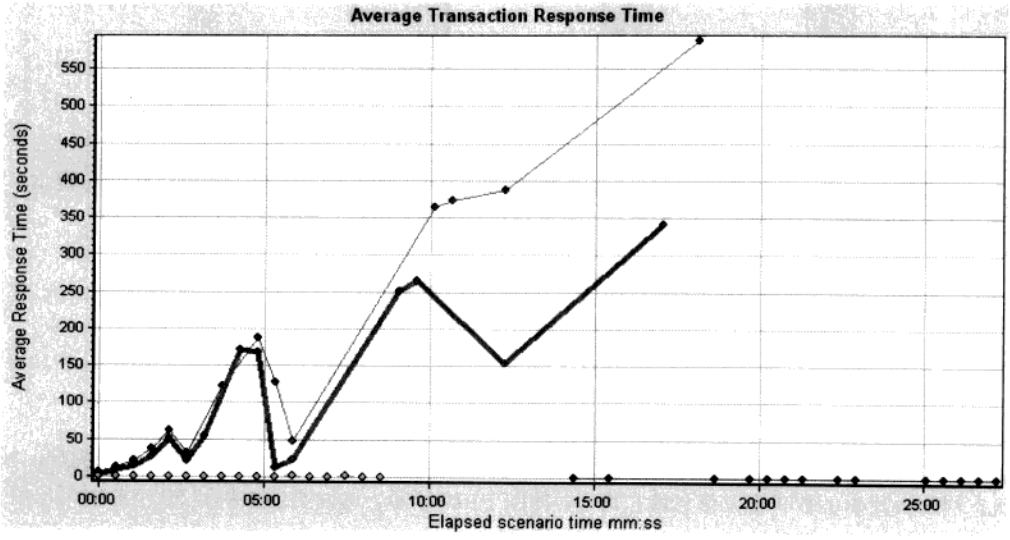


图 10-30 平均事务响应时间

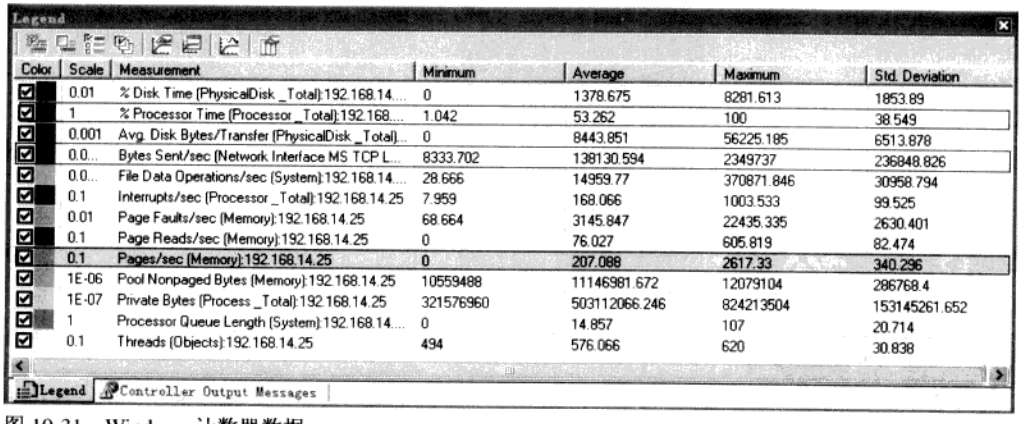
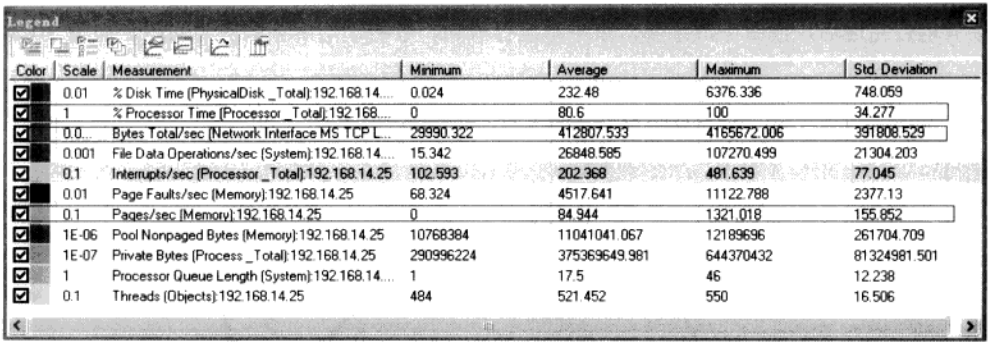


图 10-31 Windows 计数器数据

要解决这个问题，必须优化系统配置，否则系统无法处理 100 个虚拟用户并发登录。在这里是因为进行实例讲述，不方便去优化系统配置，但可以降低并发用户数，测试 35 个虚拟用户并发的情况。

在 35 个虚拟用户时，CPU 的使用率还是几乎达到 100%，内存 Pages/sec 的指标超过 80，如图 10-32 所示。说明硬件配置在 35 个虚拟用户并发登录时，还不是最好的，但是并没有失败的事务，这说明处理 35 个虚拟用户并发登录应用服务器时没有任何问题。

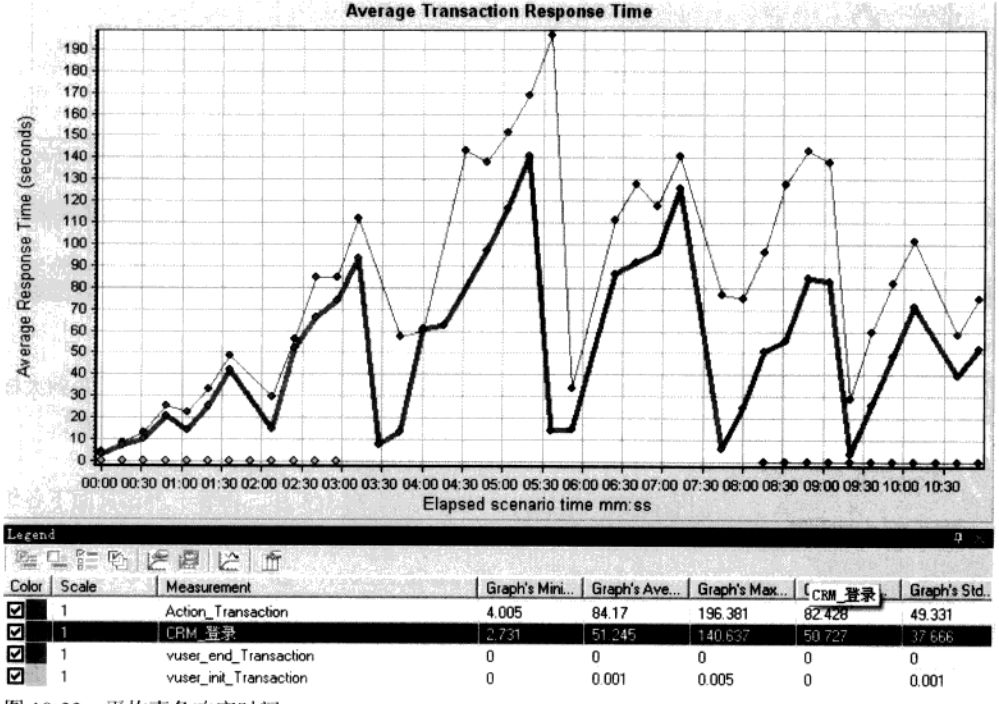


Legend

Color	Scale	Measurement	Minimum	Average	Maximum	Std. Deviation
☑	0.01	% Disk Time (PhysicalDisk _Total):192.168.14....	0.024	232.48	6376.336	748.059
☑	1	% Processor Time (Processor _Total):192.168....	0	80.6	100	34.277
☑	0.0...	Bytes Total/sec (Network Interface MS TCP L...	29990.322	412807.533	4165672.006	391808.529
☑	0.001	File Data Operations/sec (System):192.168.14....	15.342	26848.585	107270.499	21304.203
☑	0.1	Interrupts/sec (Processor _Total):192.168.14.25	102.593	202.368	481.639	77.045
☑	0.01	Page Faults/sec (Memory):192.168.14.25	68.324	4517.641	11122.788	2377.13
☑	0.1	Pages/sec (Memory):192.168.14.25	0	84.944	1321.018	155.852
☑	1E-06	Pool Nonpaged Bytes (Memory):192.168.14.25	10768384	11041041.067	12189696	261704.709
☑	1E-07	Private Bytes (Process _Total):192.168.14.25	290996224	375369649.981	644370432	81324981.501
☑	1	Processor Queue Length (System):192.168.14....	1	17.5	46	12.238
☑	0.1	Threads (Objects):192.168.14.25	484	521.452	550	16.506

图 10-32 Windows 计数器数据

但是我们发现平均事务响应太长，如图 10-33 所示。而手动去登录该系统时，也同样感觉到很慢。



Legend

Color	Scale	Measurement	Graph's Mini...	Graph's Ave...	Graph's Max...	CRM_登录	Graph's Std.
☑	1	Action_Transaction	4.005	84.17	196.381	82.428	49.331
☑	1	CRM_登录	2.731	51.245	140.637	50.727	37.666
☑	1	vuser_end_Transaction	0	0	0	0	0
☑	1	vuser_init_Transaction	0	0.001	0.005	0	0.001

图 10-33 平均事务响应时间

在这里就不再继续测试下去了，有兴趣的朋友可以自己试试。通过上面的结果可以得知，系统无法处理 100 个虚拟用户同时并发登录，在使用 50 个虚拟用户并发登录时，虽然没有失败的事务，但是事务的处理时间过长，平均时间大概在 60 秒。这是因为服务器的硬件配置引起的。

2. 进入联系人管理界面

场景运行完成后，没有失败事务，事务都运行成功。分析平均事务响应时间图，如图 10-34 所示。

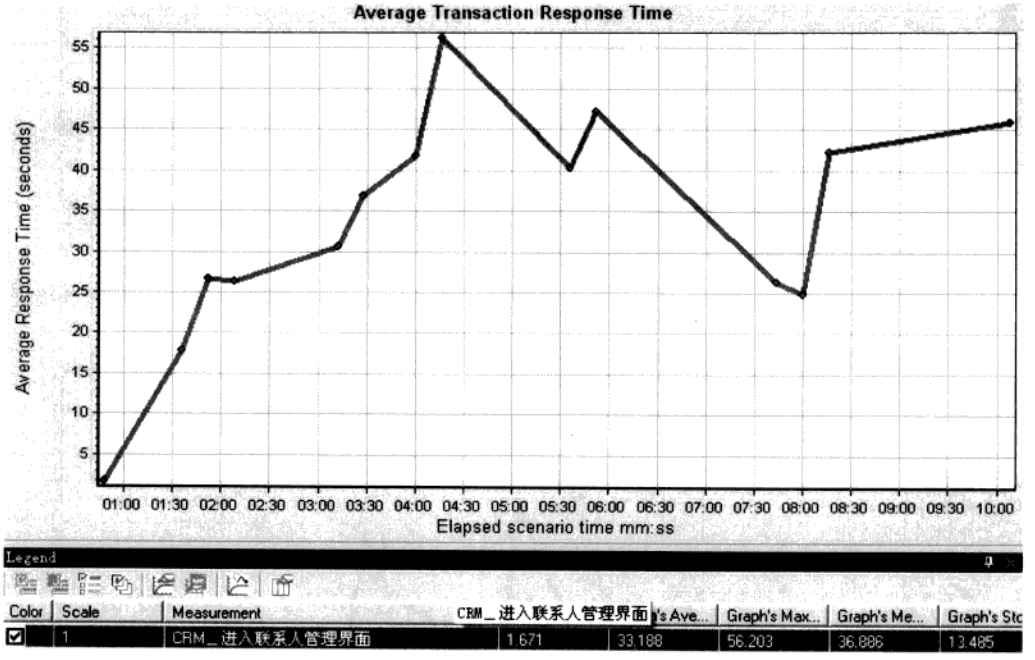


图 10-34 平均事务响应时间

从上图可以看到平均时间在 33.188 秒，这个时间应该不是很确切，从曲线图中可以看到大概是在 40 秒，但不管哪个时间，这个平均时间显示是太长了，这样的系统性能不能让人满意。

接着通过页面细分的方法来看一下为什么事务的响应时间这么长，如图 10-35 所示。

从这里不难看出，这个页面所花的时间太长了，最大达到 28.313 秒，再看一下该页面下各组件所花的时间，如图 10-36 所示。

通过这个图可以看到http://192.168.14.25/themes/Sugar/images/print.gif所花的时间最长，并且还有报错的现象。但通过上面这个图还是无法确定是网络存在问题还是服务器存在问题，为了确定这个问题，进一步分析“Time to First Buffer（Over Time）”的情况，如图 10-37 所示。

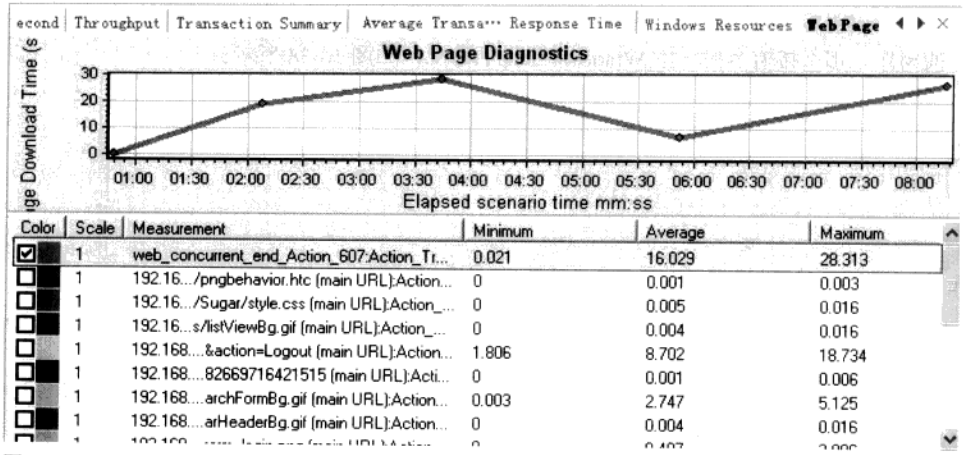


图 10-35 页面细分

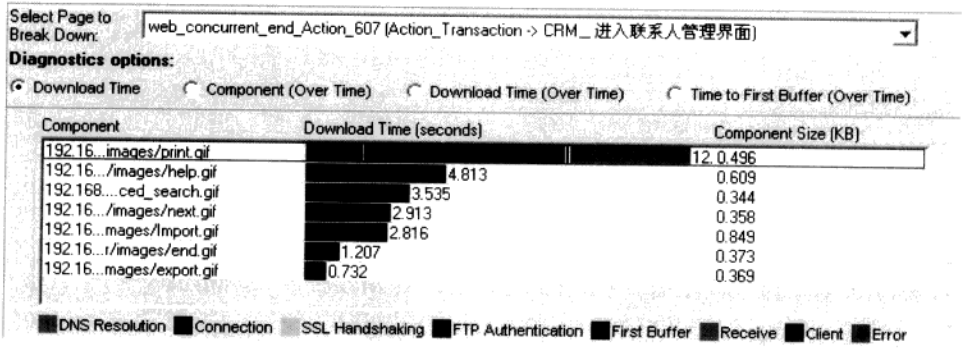


图 10-36 各组件的响应时间

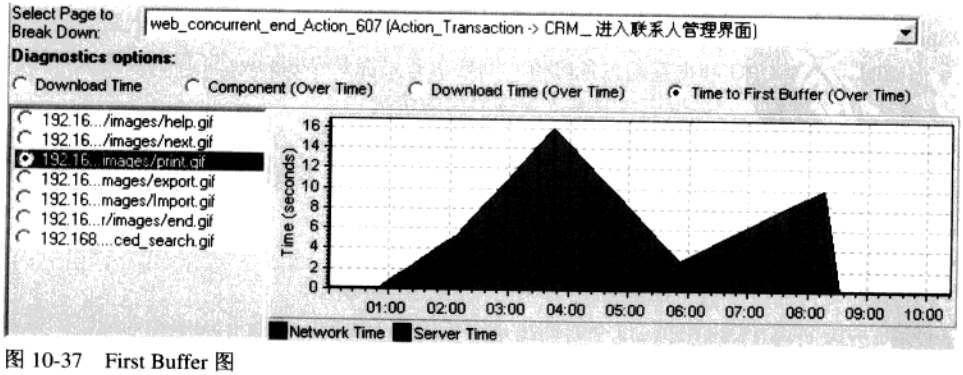
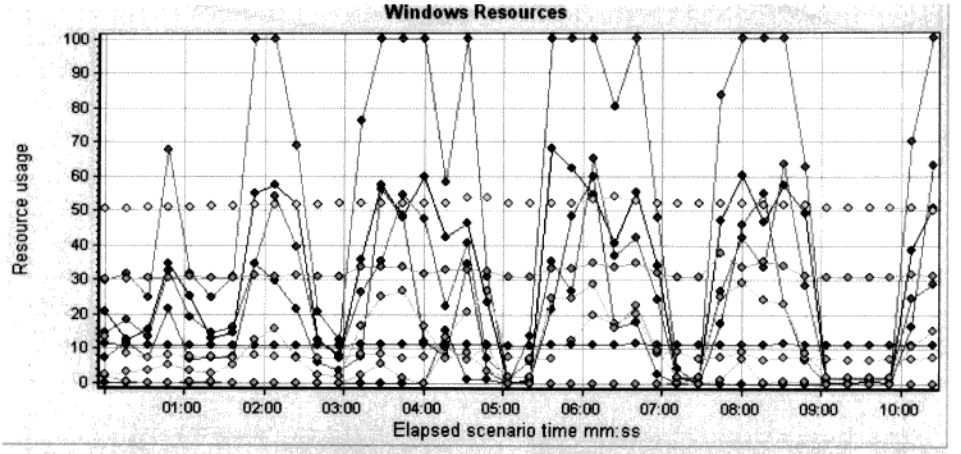


图 10-37 First Buffer 图

通过这个图很明显地看到是由于网络时间太长导致的，但是按局域网的带宽来看，网络应该不会有很大的问题。那么接着分析一下 Windows 资源的情况，如图 10-38 所示。



Legend

Color	Scale	Measurement	Minimum	Average	Maximum
☑	0.1	% Disk Time (PhysicalDisk _Total):192.168.14....	0.009	5.459	918.957
☑	1	% Processor Time (Processor _Total):192.168....	0	55.568	100
☑	0.0...	Bytes Total/sec (Network Interface MS TCP L...	16384.034	249462.168	1584128.642
☑	0.001	File Data Operations/sec (System):192.168.14....	7.99	17701.976	89219.773
☑	0.1	Interrupts/sec (Processor _Total):192.168.14.25	45.802	98.494	457.849
☑	0.01	Page Faults/sec (Memory):192.168.14.25	35.666	3221.738	10568.421
☑	0.1	Pages/sec (Memory):192.168.14.25	0	13.62	634.308
☑	1E-06	Pool Nonpaged Bytes (Memory):192.168.14.25	11190272	11341004.8	12034048
☑	1E-07	Private Bytes (Process _Total):192.168.14.25	301088768	322348158.781	440954880
☑	1	Processor Queue Length (System):192.168.14....	1	11.329	39
☑	0.1	Threads (Objects):192.168.14.25	497	519.981	553

图 10-38　Windows 资源图

在 Windows 资源图中可以看到网络传输 Bytes Total/sec 的最大值是 2.49M，而我们网络的带宽有 100M。但同时也发现 CPU 和内存超过危险值，同样也有可能是 Windows 资源引起请求时间过长，进一步测试 20 个虚拟用户并发进入联系人管理界面的情况。

场景运行结束后，可以很明显的看出，平均事务响应时间在 5 秒，通过页面细分发现网络占用的时间已经变成了 3 秒，如图 10-39 所示。这说明在 25 个虚拟用户并发时的网络传输时间是由于 Windows 资源引起的。

3. 新增联系人

场景运行完成后，首先分析平均事务响应时间，如图 10-40 所示。

平均事务响应时间为 15.893 秒，这个时间也是很长的一个时间，并且平均事务响应时间像波浪一样有规律的出现。进一步查看页面细分图，如图 10-41 所示。

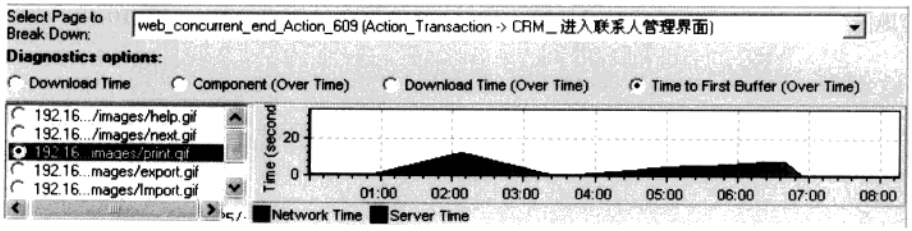


图 10-39 First Buffer 图

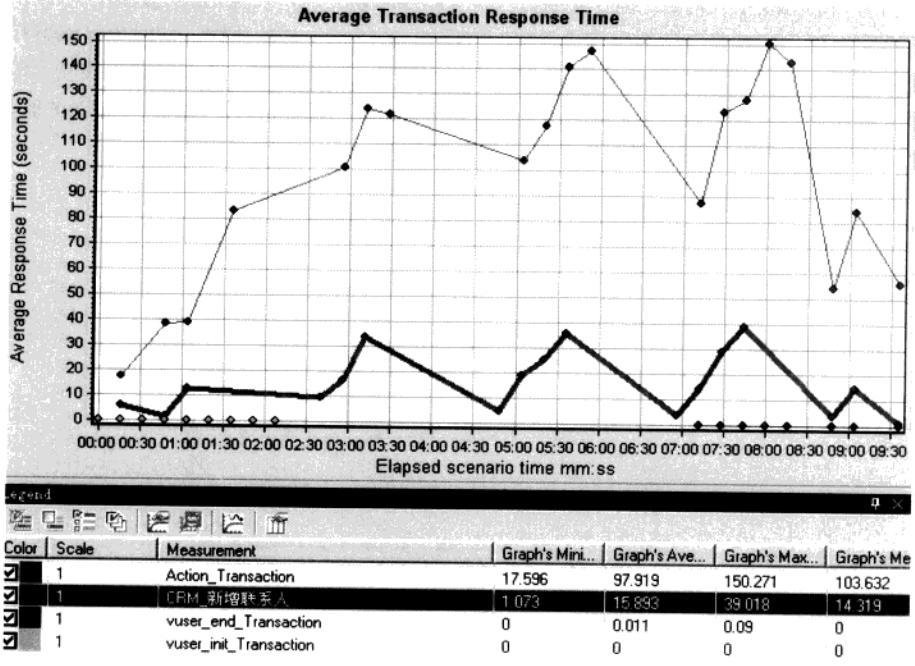


图 10-40 增加联系人信息平均事务响应时间

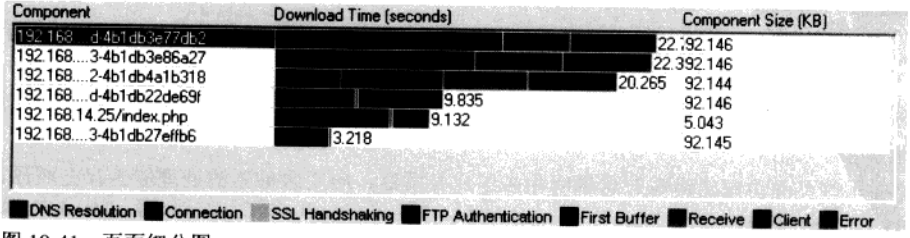


图 10-41 页面细分图

可以明显地看到访问那几个页面的时间长短，看一下这个页面显示的内容，如图 10-42 所示。

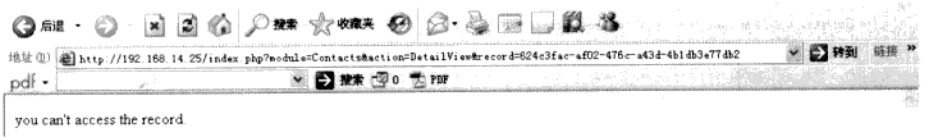


图 10-42 页面查看

结果通过这个页面无法看到记录，该页面是在新增加一条联系人记录并保存后进入的页面，本来应该是显示当前添加的联系人信息，但在实验过程中，添加联系人后，系统无法从数据库中读出该数据，导致页面出错，这就是为什么平均事务响应时间达到 15.893 秒的原因，同时也可以解释为什么平均事务像波浪一样。因为在新增联系人时，这个页面有时访问成功，有时访问不成功。但仅仅通过这个来解释还不是很完全，下面将平均事务响应时间图和 Windows 资源图进行自动关联，来查看平均事务响应时间与 Windows 资源图的关联度，如图 10-43 所示。

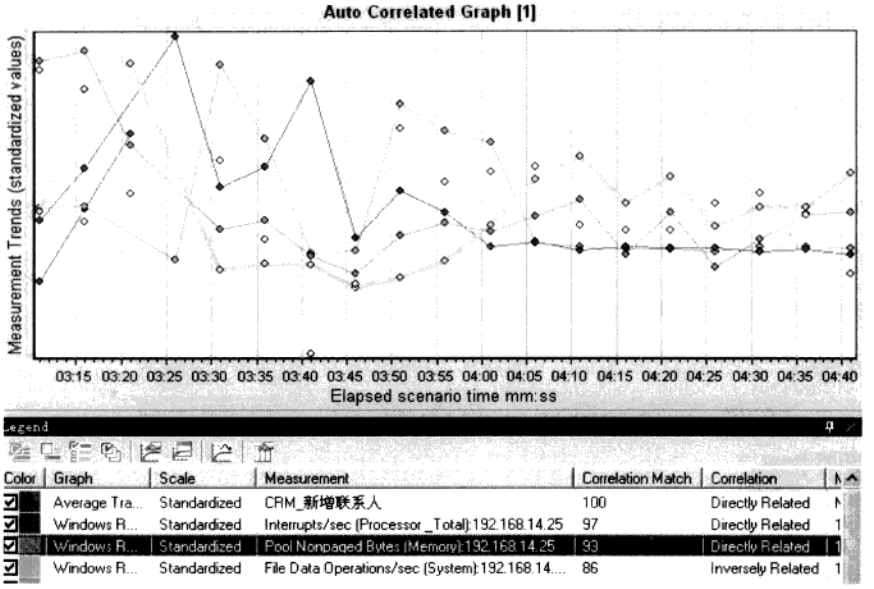


图 10-43 自动关联图

通过自动关联可以看到当内存使用率上升时，事务的响应时间也在增长。所以平均事务响应时间图与系统资源也有关系。

那么为什么一些页面访问出错呢？这说明在多用户并发后，应用服务器的处理能力存在瓶颈，但这也很有可能是系统配置引起的。

接下来进行这样的实验，使用 20 个虚拟用户并发进行新增联系人操作，并将联系人中现有的

记录删除掉。

结果平均事务响应时间为 6.352 秒，但是还有一些页面访问出错，如图 10-44 所示。

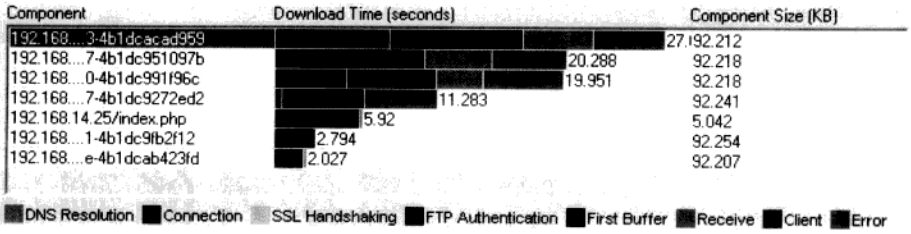


图 10-44 页面细分图

通过这个实验说明应用服务器在处理并发事务的时候还存在问题，特别是内存的使用方面。而应用服务器 Apache 和数据库 Mysql 出现的问题应该是比较小的。

还有其他的几个模块这里就不一一讲述了，有兴趣的朋友可以自己进行测试一下。

10.7 测试结论

每个模块的测试结果如下：

1. 登录

登录模块，服务器当前的配置无法处理 100 个虚拟用户并发的活动，测试 50 个虚拟用户并发时，发现事务都能被成功的处理，但是登录的时间过长，平均时间为 60 秒，系统资源也超过安全指标，但应用服务器正常，可以通过优化服务器的配置来提高性能。

2. 进入联系人管理界面

进入联系人管理界面模块，在 25 个虚拟用户并发时，平均事务响应时间在 40 秒，同时网络传输时间明显过长，系统资源使用超过安全指标，再试验 20 个虚拟用户并发，发现平均事务响应时间为 5 秒，网络传输也正常。这说明通过调优系统配置可以提高性能。

3. 增加联系人

25 个虚拟用户并发，平均事务响应时间为 15.893 秒，服务器无法处理 25 个虚拟用户并发，有页面访问出错的现象，主要可以通过调节系统配置来优化性能，与应用服务器、数据库服务器的关系不大。

其他的几个模块也是大同小异，在此不进行详细的分析。

注：在进行结果判断时有一个小技巧，当无法确定平均事务时间过长是由系统配置引起时，可以在运行脚本期间，手动测试该业务，如果手动测试该业务时还是一样缓慢，那说明服务器的配置有问题，而不是脚本或其他的原因引起的。在 Windows 资源监控的过程中可以将 LoadRunner 显示的数据和 Windows 操作系统中自带的性能监控对比，查看监控的数据是否准确，这就是为什么在测试过程对服务器端的性能指标进行监控的原因之一。

11 信息系统性能测试

11.1 系统介绍

该系统是 C/S 模式架构，开发环境为 Visual Studio 2005，使用.NET 2.0 套件进行开发，使用的数据库为 SQL Express 2005，应用服务器为 IIS。系统架构图如图 11-1 所示。

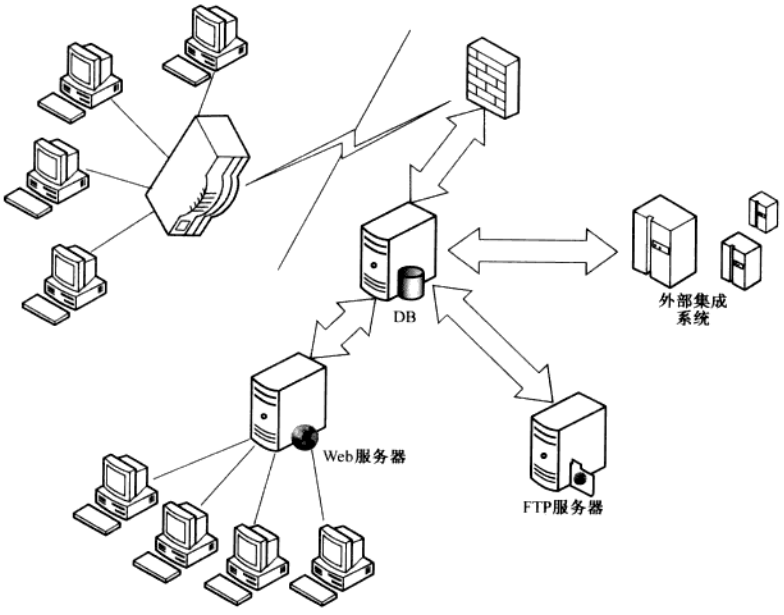


图 11-1 系统架构图

其主要功能包括：数据传输、检索和数据查看（包括修改、删除）。

1）数据传输：客户端将收集到的数据传输到服务器，使用的协议为 FTP 文本传输协议。

2）检索：设置不同的条件检索需要的数据。

3）新增记录：用户可以自己新增需要的记录。

11.2 需求分析

11.2.1 性能指标

分析性能指标，将客户需求与本系统相结合，用户希望以下模块能满足以下性能指标，见表 11-1。

表 11-1 用户对系统性能指标的要求

指标种类		用户需求
登录		至少支持 15 个用户同时并发登录，登录的响应时间不能超过 5 秒
业务	数据检索	15 个用户同时检索，检索数据不能超过 30 秒
	新增记录	至少支持 15 个用户同时新增记录，新增一条记录信息的时间不能超过 5 秒

如果仅仅根据表 11-1 提供的性能需求进行有效的性能测试，确实是一件很难的事情，必须在现有的需求基础上再进行细化，提练出一份可以指导性能测试的需求。

需要说明的一点是，因为本书只是使用该系统来作为对性能测试实践分析的一个案例，故只选择了部分模块进行测试，并没有对整个系统性能测试工作进行全面的描述。

11.2.2 需求详细分析

该系统要求至少能保存 6 年的数据，而根据现在的情况，每天传输上去的数据至少在 200 个，这样的话，6 年后系统中至少有 43 万个数据，故设定测试环境是系统已经存在 50 万个数据。目前使用的用户数只有 20 个，即使 6 年后，使用该系统的人数也不会超过 50 个。各模块的需求分析如下：

1. 登录

这里原始需求中并没有对系统的环境进行介绍，而系统现有数据量的多少很有可能会对系统登录的时间带来影响。通过对历史数据的分析可以确定系统必须在已经存在 50 万个数据的情况下进行测试，测试 15 个用户并发登录，登录的响应时间不能超过 5 秒。

2. 数据检索

原始需求中没有对检索的环境进行介绍，根据登录模块的分析情况，确定在系统已存在 50 万个数据的情况下进行检索，但是检索多少个数据呢？原始需求并没有相关的介绍，因为根据检索结果不一样，检索过程的响应时间也是不一样的，所以这里定义检索结果为 100 个数据的情况。

3. 新增记录

同前面两个模块的原始需求一样，该模块需求也没有介绍在什么样的系统环境下进行新增记录

操作，当然这里也定义为系统已经存在 50 万个数据的情况下进行新增记录操作。但这里新增记录包含两层意思：进入新增记录界面和新增记录两个过程，必须测试 15 个用户并发进入新增记录界面的时间以及提交一个新增记录的时间。

需求细化后的情况，见表 11-2。

表 11-2 详细化后的需求

指标种类		用户需求
登录		在系统已经存在 50 万个数据的情况下，至少支持 15 个用户同时并发登录，登录的响应时间不能超过 5 秒
业务	数据检索	在系统已经存在 50 万个数据的情况下，15 个用户同时检索，检索结果为 100 条数据，检索数据不能超过 30 秒
	进入新增记录界面	在系统已经存在 50 万个数据的情况下，至少支持 15 个用户同时并发进入新增记录界面，进入新增记录界面的时间不能超过 5 秒
	提交新增记录	在系统已经存在 50 万个数据的情况下，至少支持 15 个用户同时并发新增记录，新增一条记录信息的时间不能超过 5 秒

11.3 测试方案及计划

11.3.1 人力资源

性能测试作为软件测试的一部分工作，一般都是在系统测试完成后，或者是系统功能比较稳定对性能测试没有影响的情况下进行的。根据测试计划，性能测试允许的时间为 25 个工作日，故计划由 1 个人进行测试。

11.3.2 时间进度

性能测试的计划和时间进度安排，见表 11-3。

表 11-3 性能测试计划

性能测试	25 工作日	2009 年 6 月 8 日	2009 年 7 月 10 日
性能测试用例设计	4 工作日	2009 年 6 月 8 日	2009 年 6 月 11 日
测试环境搭建	3 工作日	2009 年 6 月 12 日	2009 年 6 月 16 日
测试数据准备	3 工作日	2009 年 6 月 17 日	2009 年 6 月 19 日
脚本开发	5 工作日	2009 年 6 月 22 日	2009 年 6 月 26 日
测试执行	5 工作日	2009 年 6 月 29 日	2009 年 7 月 3 日
测试结果分析	5 工作日	2009 年 7 月 6 日	2009 年 7 月 10 日

因为项目的时间比较短，在这里就不再做里程碑图。

11.3.3 测试环境准备

在进行测试前，必须先搭建好测试平台。

服务器安装操作系统为 Windows 2003 系统，其中数据库服务器和应用服务器安装在同一台机器上，服务器的 IP 地址为 192.168.8.56。

测试机安装的操作系统为 Windows XP 系统，因为测试的并发用户数最多为 15 个，故只要一台测试机即可，其中 Controller 和负载机为同一台机器。测试机与服务器在同一个局域网内。

在这里有一个需要注意的地方就是，客户端的机器不能配置太差，否则会影响事务的响应时间，因为该系统是一个 C/S 模式，而 C/S 模式架构的系统有一个特点就是在运行的时候，服务器会将部分压力转化到客户端机器，从而减轻客户端对服务器的压力，所以在配置测试环境的时候不能让测试机配置太差，否则测试机就会成为性能的瓶颈。

详细的配置见表 11-4。

表 11-4 测试机与服务器软硬件配置

设备	硬件配置	软件配置
数据库服务器 应用服务器 FTP 服务器	PC 机（一台） CPU：Intel Xeon X3200 2.4GHz 内存：2.0GB 硬盘：300GB	Windows 2003 SQL Express 2005 IIS FTP
控制器 负载机	PC 机（一台） CPU：Intel Celeron 3.06GHz 内存：512MB 硬盘：80GB	Windows XP LoadRunner 9.1 IE 6.0 Microsoft Office

测试拓扑结构图，如图 11-2 所示。

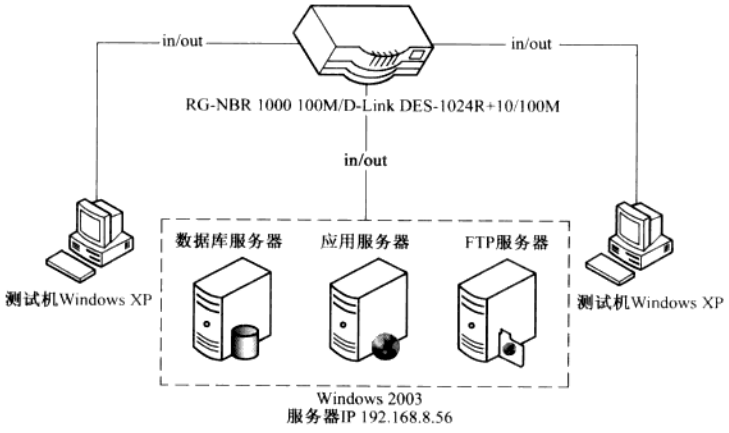


图 11-2 测试拓扑结构图

测试工具：LoadRunner 9.1

录制协议：Web Services

11.3.4 业务模型创建

在前面的章节中已经对业务模式进行了相关介绍，该案例业务模型见表 11-5。

表 11-5 业务模型

指标种类		业务模型
登录		1. 准备 15 个具有权限的用户 2. 15 个虚拟用户同时并发进行登录操作
业务	数据检索	1. 为系统准备 50 万个数据 2. 设置好检索条件，保证检索的结果在 100 个数据左右 3. 在检索界面进行检索 4. 同时进行检索的虚拟用户为 15 个
	进入新增记录界面	1. 为系统准备 50 万个数据 2. 进入新增记录界面 3. 同时进入新增记录界面的虚拟用户为 15 个
	提交新增记录	1. 为系统准备 50 万个数据 2. 进入新增记录界面 3. 提交一条新增记录 4. 同时新增记录的虚拟用户为 15 个

对数据检索业务进行进一步分析，为什么业务模式中一定要定义好检索结果呢？因为输入检索条件后，检索到的数据量的多少会直接影响到检索时间。其实这里的检索时间（T）是由两部分时间组成的：第一部分是在数据库中检索出与检索条件一致的数据（T1）；第二部分是将检索出来的数据显示到数据列表（T2），如图 11-3 所示，T=T1+T2。

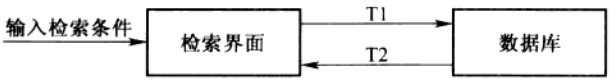


图 11-3 T=T1+T2

11.3.5 场景模型创建

在前面的章节中已经对场景模式进行了相关介绍，该案例场景模型见表 11-6。

表 11-6 场景模型

指标种类		场景模型
登录		1. 启用脚本中的集合点 2. 每 5 秒加载一个虚拟用户，虚拟用户加载完成后，持续运行 5 分钟，结束后，每 5 秒钟释放一个虚拟用户 3. 使用 IP 欺骗技术，IP 欺骗新建 15 个 IP 地址 4. 添加 Windows 计数器 5. 监控虚拟用户运行日志文件
业务	数据检索	1. 启用脚本中的集合点 2. 每 5 秒加载一个虚拟用户，虚拟用户加载完成后，持续运行 5 分钟，结束后，每 5 秒钟释放一个虚拟用户 3. 使用 IP 欺骗技术，IP 欺骗新建 15 个 IP 地址 4. 添加 Windows 计数器 5. 监控虚拟用户运行日志文件
	进入新增记录界面	1. 启用脚本中的集合点 2. 每 5 秒加载一个虚拟用户，虚拟用户加载完成后，持续运行 5 分钟，结束后，每 5 秒钟释放一个虚拟用户 3. 使用 IP 欺骗技术，IP 欺骗新建 15 个 IP 地址 4. 添加 Windows 计数器 5. 监控虚拟用户运行日志文件
	提交新增记录	1. 启用脚本中的集合点 2. 每 5 秒加载一个虚拟用户，虚拟用户加载完成后，持续运行 5 分钟，结束后，每 5 秒钟释放一个虚拟用户 3. 使用 IP 欺骗技术，IP 欺骗新建 15 个 IP 地址 4. 添加 Windows 计数器 5. 监控虚拟用户运行日志文件

11.3.6 测试数据准备

完成以上工作后，接下来就要为业务模型准备数据。需要准备的数据如下：

1. 登录用户账号

为了满足虚拟用户的需求，这里需要准备 15 个虚拟用户，并且这 15 个虚拟用户的权限要一致，以保证新增数据业务顺利进行。

2. 准备 50 万个数据

为满足业务流程中对系统环境的需求，需要准备 50 万条记录。在这里数据的生成使用了一个由开发工程师开发的小工具，该工具是通过 SQL 语言来实现的。

各模块数据准备详细情况，见表 11-7。

表 11-7 准备数据

指标种类		准备数据
登录		1. 准备好 15 个权限一致的用户账号信息 2. 准备 50 万条记录
业务	数据检索	1. 准备好 15 个权限一致的用户账号信息 2. 准备 50 万条记录
	进入新增记录界面	1. 准备好 15 个权限一致的用户账号信息 2. 准备 50 万条记录
	提交新增记录	1. 准备好 15 个权限一致的用户账号信息 2. 准备 50 万条记录

11.4 测试用例

完成业务模型和场景模型后，即可以开始设计测试，各模块测试用例设计如下：

1. 登录

用例编号：LI_001

测试目的：测试 15 个虚拟用户并发时，系统登录的响应时间。

并发用户数：15 个。

模拟用户行为：

1）进入登录界面。

2）输入用户名和密码，点击“登录”按钮。

预期结果：系统登录的响应时间不能超过 5 秒。

2. 数据检索

用例编号：QU_001

测试目的：测试检索 100 个记录，系统检索的响应时间。

并发用户数：15 个。

模拟用户行为：

1）进入登录界面。

2）输入用户名和密码。

3）进入检索界面，并设置好检索条件。

4）点击“检索”按钮。

预期结果：系统检索 100 条记录的响应时间不能超过 30 秒。

3. 进入新增记录界面

用例编号：ADD_001

测试目的：测试进入新增记录界面所需要的时间。

并发用户数：15 个。

模拟用户行为：

1）进入登录界面。

2）输入用户名和密码。

3）进入数据管理界面。

4）点击“新增”按钮，进入新增记录界面。

预期结果：系统进入新增记录界面的响应时间不能超过 5 秒。

4. 提交新增记录

用例编号：ADD_002

测试目的：测试提交一个新增记录所需要的时间。

并发用户数：15 个。

模拟用户行为：

1）进入登录界面。

2）输入用户名和密码。

3）进入数据管理界面。

4）点击“新增”按钮，进入新增记录界面。

5）新增一条记录并提交。

预期结果：系统提交一条新增记录的响应时间不能超过 5 秒。

11.5 执行测试

11.5.1 脚本开发

本节主要讲述测试脚本实现过程和脚本的结构。虚拟用户脚本的开发情况见表 11-8。

表 11-8 虚拟用户脚本开发情况

用例编号	用例名称	开发情况
LI_001	并发登录	在脚本中对用户名和密码进行参数化，参数调用是直接读取记事本中的数据，设置文本检查点，插入开始与结束事务点
QU_001	数据检索	将脚本和 LI_001 合并，在 LI_001 登录后，设置好检索条件进行数据检索，插入开始与结束事务点
ADD_001	进入新增记录界面	将脚本和 LI_001 合并，在 LI_001 登录后，在数据管理界面，点击“新增”按钮进入新增记录界面，插入开始与结束事务点
ADD_002	提交新增记录	将脚本和 LI_001 合并，在 LI_001 登录后，在数据管理界面，点击“新增”按钮进入新增记录界面之后提交一条新增记录，插入开始与结束事务点

在进行脚本录制之前首先分析在脚本开发过程中可能遇到的问题，之后再进行脚本开发过程的分析。

主要存在以下两个问题：

（1）跨网段 IP 欺骗

在脚本开发过程中，遇到的第一个问题就是如何进行 IP 欺骗，上一个案例测试机与被测试的服务器是在同一个网段，这样只要进行正常的 IP 欺骗设置即可，但现在的情况是测试机与被测试的服务器不在同一个网段，跨网段如何进行 IP 欺骗前面章节的内容有介绍过，在此就不再详细介绍。这里只讲述如何修改 nt_routing.bat 文件，LoadRunner 机器的 IP 为 192.168.14.25，使用的 IP 欺骗地址为 192.168.14.69～192.168.14.79 这 11 个 IP。下面看一下修改后的 nt_routing.bat 文件内容，如图 11-4 所示。

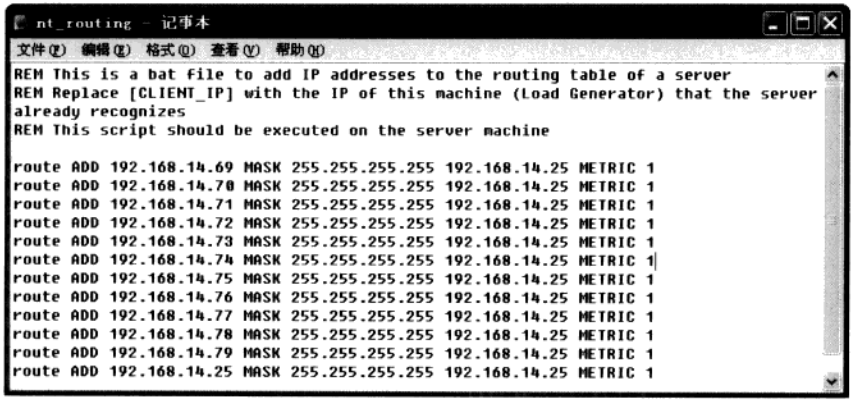


图 11-4　nt_routing 文件内容

修改完成后，在被测试的服务器上运行该文件即可。

（2）录制时无法调用应用程序

在脚本录制过程中遇到的另一个问题是，在录制脚本的时候，发现 LoadRunner 不会将要录制的应用程序调用出来，并且录制工作栏显示的 events 一直为零，如图 11-5 所示。

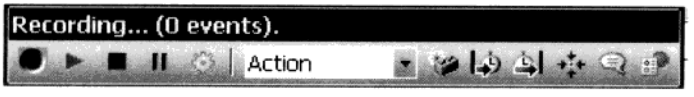


图 11-5　录制的事务一直为 0

找了很久一直没有找到原因所在，因为正常情况即使录制时的事务为零，LoadRunner 也一定会将要录制的应用程序调用出来。经过多次实验还是一样无法录制，然后与开发工程师就开发环境进行沟通。当时开发工程师说明使用的是.NET 2.0 的插件，但并没有意识到这个插件会存在问题。因为在潜意识里，一直认为 LoadRunner 只和协议有关，与开发语言没有任何关系，也不存在插件

的问题，只是 QTP 存在插件的问题。在网上找 LoadRunner 插件的相关资料，发现一个这样的问题，LoadRunner 有两个插件：.NET 和 JBuilder。当时使用的是 LoadRunner 8.1 版本，LoadRunner 8.1 版本使用的是.NET 1.0 的插件，而开发使用的是.NET 2.0 的插件，但当时还不能完全确定是由这个原因引起的，就下载了 LoadRunner 9.1 版本进行试验（因为 LoadRunner 9.1 版本是.NET 2.0 的插件），试验成功了，能正确的录制，问题解决了。当然还有一个解决办法就是重新安装一个最新插件即可。

解决了上面两个问题后，即可开始录制脚本。

1. 登录

在录制脚本时，为了测试登录的事务响应时间，应该插入开始和结束事务点，并且为了保证虚拟用户并发，需要添加集合点。并发登录用例的脚本结构如图 11-6 所示。

图 11-6 登录脚本结构图

对登录的用户名进行参数化，由于这里只有 15 个虚拟用户，故参数全部放在记事本即可。密码则不需要进行参数化，因为在准备用户账号信息时，都使用一个固定的密码，也可以使用不同的密码，这并不会对测试过程有任何影响。

2. 数据检索

在录制脚本时，为了保证虚拟用户并发检索，需要添加一个检索的集合点，同时为了测试检索事务的时间，需要插入开始与结束事务。

在这里同样只需要对登录的用户名进行参数化即可。具体的脚本结构图如图 11-7 所示。

3. 进入新增记录界面

在录制脚本时，为了保证虚拟用户并发进入数据管理界面，需要添加一个检索的集合点，同时为了测试进入数据管理界面的时间，需要插入开始与结束事务。

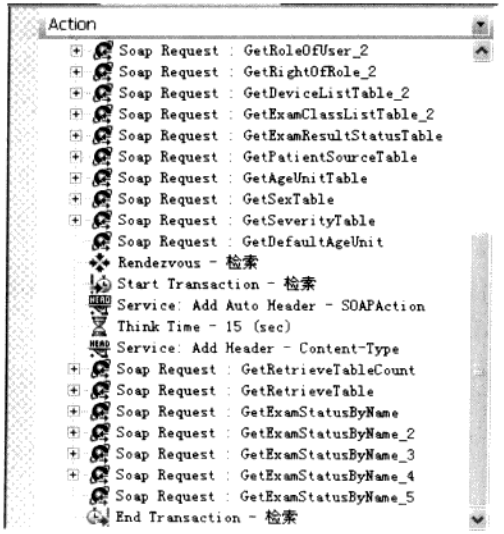


图 11-7 数据检索脚本结构图

在这里只需要对登录的用户名进行参数化即可，具体的脚本结构图如图 11-8 所示。

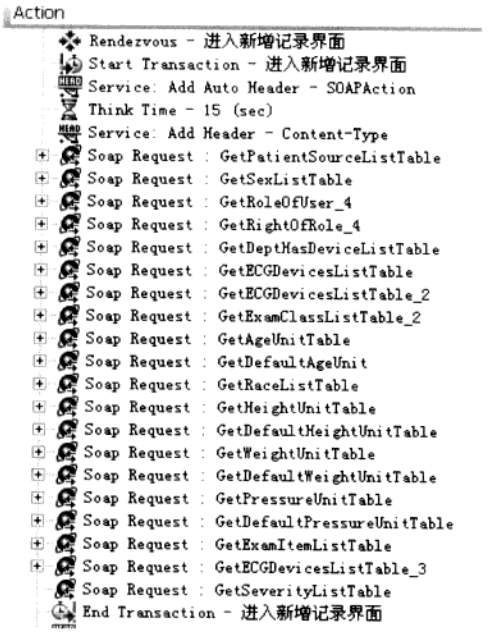


图 11-8 进入新增记录界面脚本结构图

4. 提交新增记录

在录制脚本时，为了保证虚拟用户并发新增记录，需要添加一个检索的集合点；为了测试提交新增记录的时间，需要插入开始与结束事务。

在这里不仅需要对登录的用户名进行参数化，同时为了保证新增的记录信息不一样，也必须对新增记录的内容进行参数化。具体的脚本结构图如图 11-9 所示。

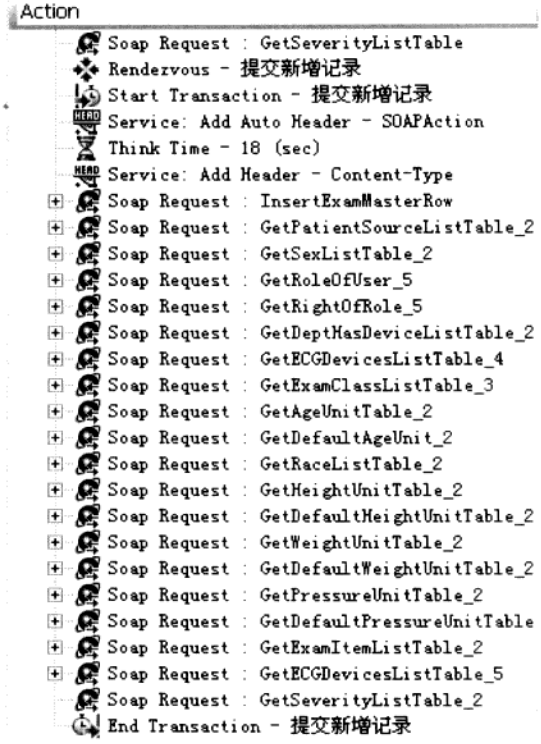


图 11-9 提交新增记录脚本结构图

需要注意的一点是，这里虽然只使用了 15 个虚拟用户进行测试，但是并不代表新增记录只有 15 条，因为在后面迭代的过程中，脚本在不停的运行，直到场景结束，这个过程一直在新增记录，所以在这里要多准备一些数据，在此准备 100 条记录。

到这里脚本开发完毕。

11.5.2 场景设计

场景设计主要是对 Controller（控制器）进行设置，设置脚本运行时的环境。各模块的场景设计情况如下：

1. 登录

场景组设置 15 个虚拟用户，如图 11-10 所示。

Group Name	Script Path	Quantity	Load Generators
登录 3	E:\ENS\登录	15	localhost

图 11-10 场景组设计

场景策略设置，在脚本运行前对所有的虚拟用户进行初始化，每 5 秒加载一个虚拟用户，虚拟用户加载完成后，连续运行 5 分钟，运行完毕后每 5 秒钟释放一个虚拟用户，如图 11-11 所示。

Action	Properties
Initialize	Initialize each Vuser just before it runs
Start Vusers	Start 15 Vusers: 1 every 00:00:05 (HH:MM:SS)
Duration	Run for 00:05:00 (HH:MM:SS)
Stop Vusers	Stop all Vusers: 1 every 00:00:05 (HH:MM:SS)

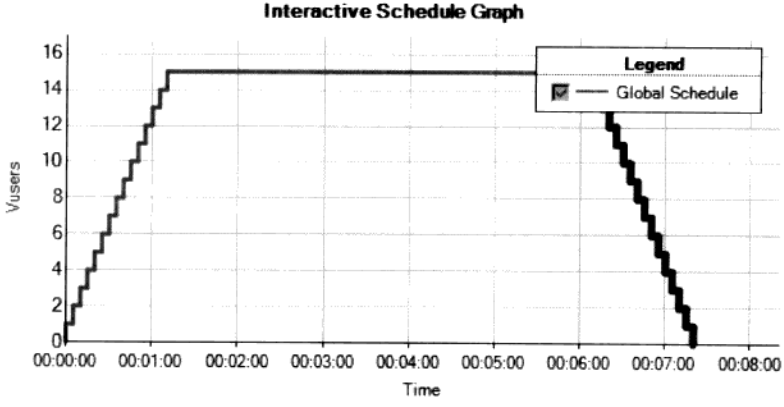


图 11-11 场景策略设置

启动 IP 欺骗功能，脚本中所有集合点都设置为使用的状态，如图 11-12 所示。

2. 数据检索

场景组设置 15 个虚拟用户，如图 11-13 所示。

场景策略设置中，在脚本开始运行之前对所有的虚拟用户进行初始化，开始虚拟用户中设置为每 5 秒加载一个虚拟用户，虚拟用户加载完成后，持续运行 5 分钟，运行结束后每 5 秒钟释放一个虚拟用户，如图 11-14 所示。

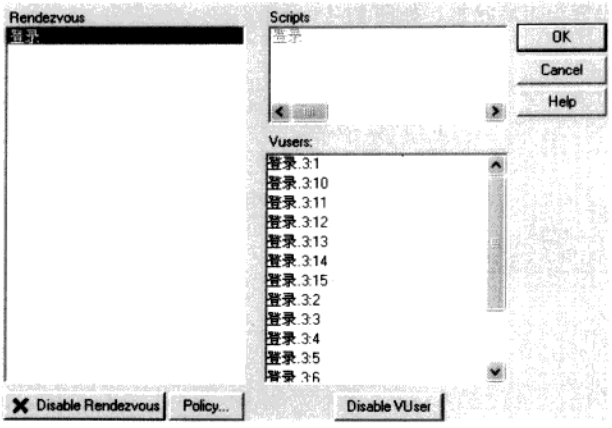


图 11-12 集合点设置

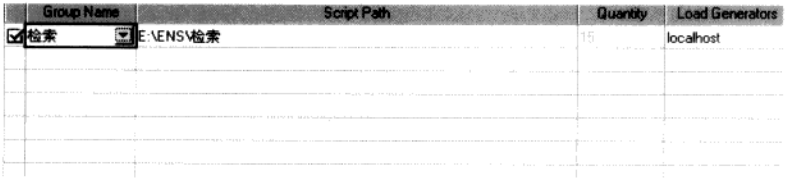


图 11-13 场景组设计

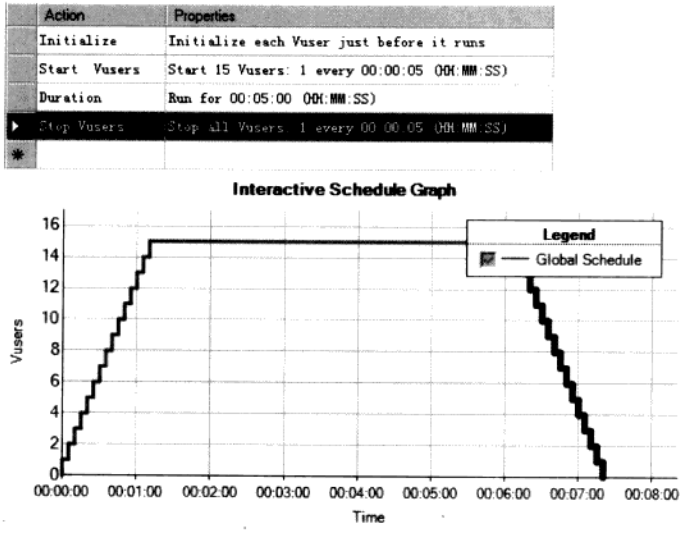


图 11-14 场景策略设置

启用 IP 欺骗，并将所有的集合点都设置为可用状态，如图 11-15 所示。

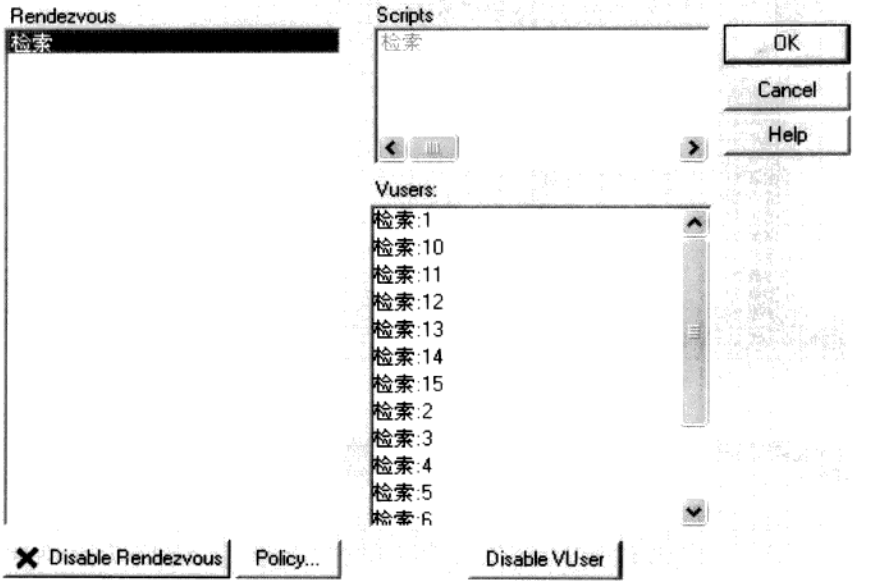


图 11-15 集合点设置

3. 进入新增记录界面

场景组中设置 15 个虚拟用户，如图 11-16 所示。

Group Name	Script Path	Quantity	Load Generators
进入新增记录界	E:\ENS\进入新增记录界面	15	localhost

图 11-16 场景组设置

场景策略设置，在脚本开始运行之前对所有的虚拟用户进行初始化，开始虚拟用户中设置每 5 秒加载一个虚拟用户，虚拟用户加载完成后，持续运行 5 分钟，运行结束后每 5 秒钟释放一个虚拟用户，如图 11-17 所示。

启用 IP 欺骗，并将所有的集合点都设置为可用状态，如图 11-18 所示。

4. 提交新增记录

场景组中设置 15 个虚拟用户，如图 11-19 所示。

	Action	Properties
	Initialize	Initialize each Vuser just before it runs
	Start Vusers	Start 15 Vusers: 1 every 00:00:05 (HH:MM:SS)
	Duration	Run for 00:05:00 (HH:MM:SS)
▶	Stop Vusers	Stop all Vusers: 1 every 00:00:05 (HH:MM:SS)
*		

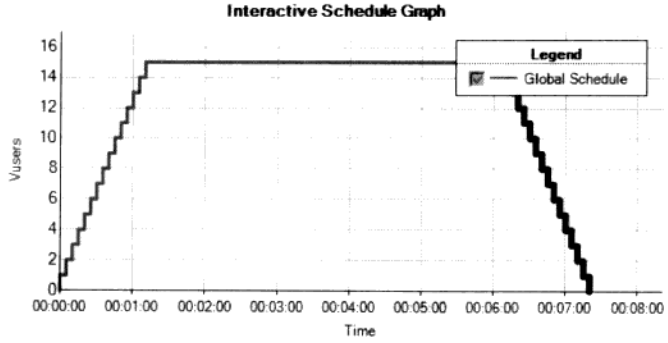


图 11-17 场景策略设置

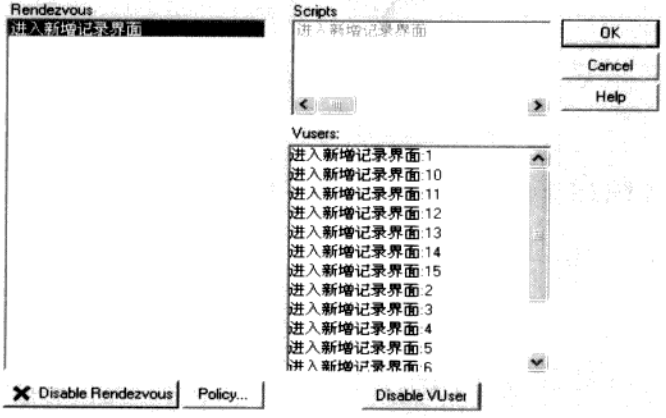


图 11-18 集合点设置

	Group Name	Script Path	Quantity	Load Generators
☑	提交新增记录	E:\ENS\提交新增记录		localhost

图 11-19 场景组设置

场景策略设置中，在脚本开始运行之前对所有的虚拟用户进行初始化，开始虚拟用户中设置每5秒加载一个虚拟用户，虚拟用户加载完成后，持续运行5分钟，运行结束后每5秒钟释放一个虚拟用户，如图11-20所示。

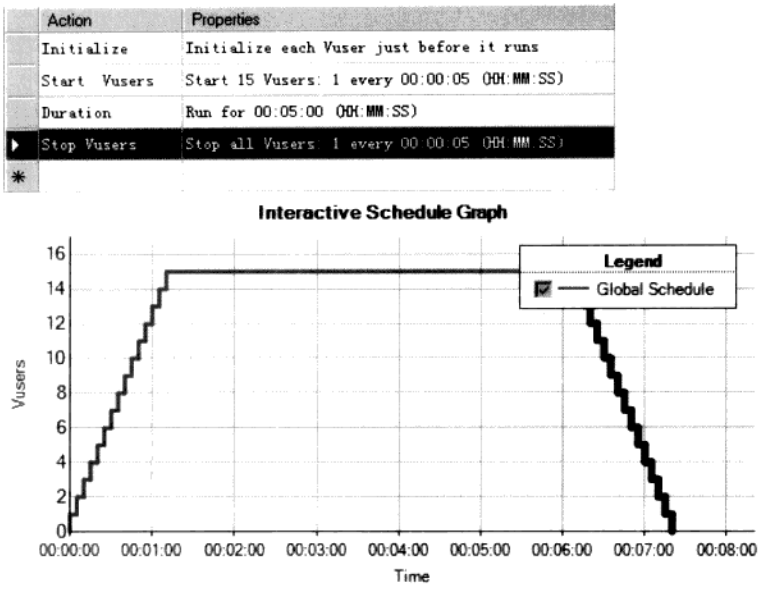


图11-20 场景策略设置

启用IP欺骗，并将所有的集合点都设置为可用状态，如图11-21所示。

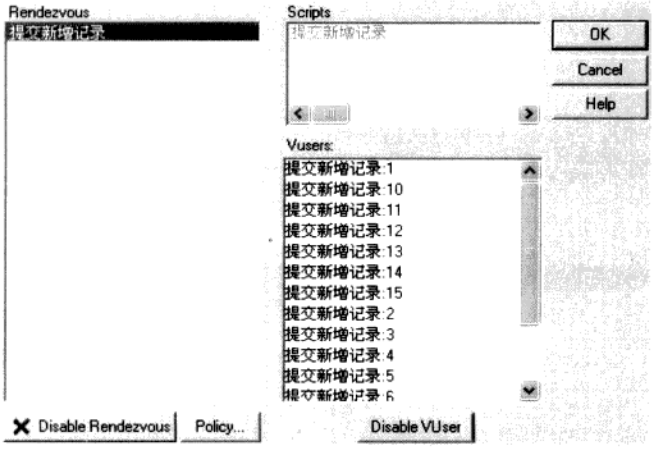


图11-21 集合点设置

到这里场景设计部分也已经完成了。

11.5.3 计数器设置

为了监控场景运行时服务器的资源使用情况，必须添加计数器，这里主要对服务器的 Windows 资源进行监控，其中这三个模块添加计数器的情况都是一致的，在这里以登录模块为例进行试验。具体如何添加计数器不做详细的描述，添加计数器后，LoadRunner 默认选择了一些监控的指标，如果有需要可以自己进行适当的添加。Windows 资源监控图如图 11-22 所示。

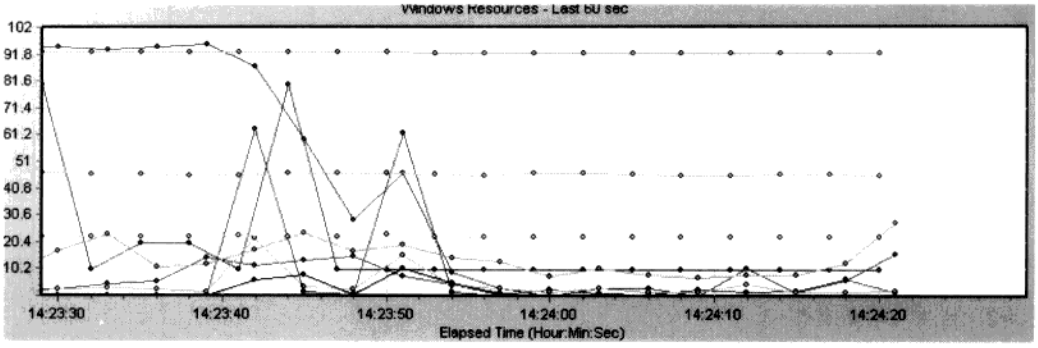


图 11-22 Windows 计数器

11.5.4 场景监控

在场景运行的过程中需要监控以下几个部分的数据信息，这三个模块在场景运行过程中需要监控的数据都是一致的。

1. 场景组运行控制信息

监控场景组中所有虚拟用户运行的情况，如图 11-23 所示。

Group Name	Script Path	Quantity	Load Generators
☑登录.3	E:\ENS\登录	15	localhost

监控虚拟用户运行的情况

图 11-23 场景组中虚拟用户运行的情况

同时还要监视虚拟用户组中每个虚拟用户运行的情况，并且一定要观察每个虚拟用户运行时的日志文件，如图 11-24 所示。

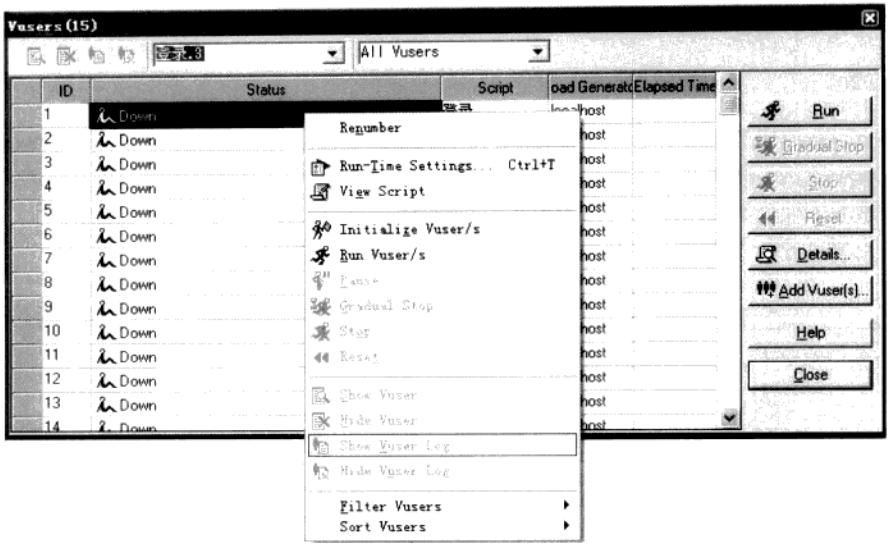


图 11-24 监视虚拟用户组运行的情况和日志文件

2. 监控场景状态图

场景状态监控，主要是监控失败和成功的事务数以及错误信息，错误信息可以帮助找到性能的瓶颈。还有一些其他的项可以关注，如图 11-25 所示。

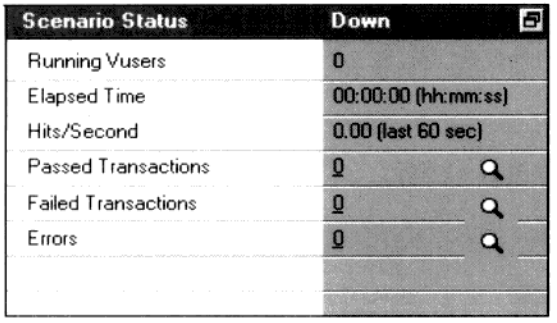


图 11-25 监视场景运行状态图

3. 监控输出对话框

当场景状态中出现错误信息时，应该通过输出对话框来查看具体的错误信息，这些错误信息有助于调试脚本和分析结果，如图 11-26 所示。

4. 监视数据图

在数据图部分主要监视 4 个数据视图的变化（正在运行虚拟用户数、事务的响应时间、每秒点击率和 Windows 计数器），如图 11-27 所示。

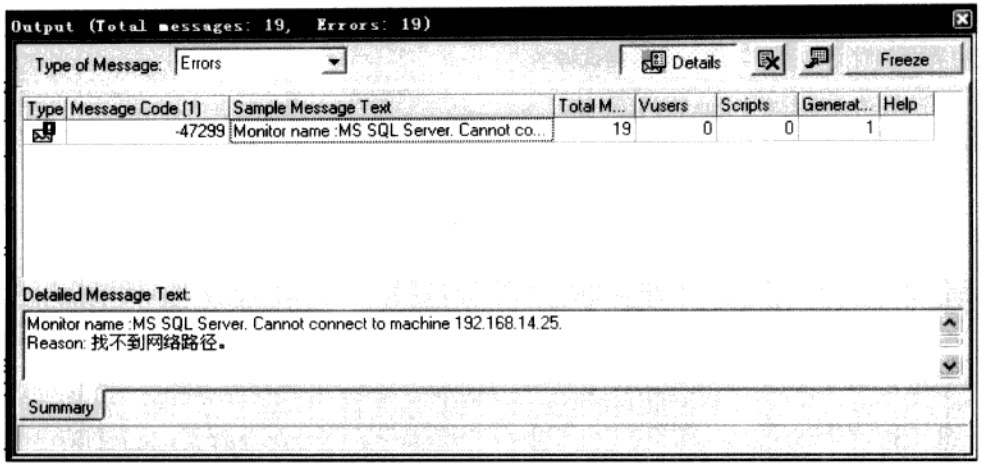


图 11-26 监视输出对话框的提示信息

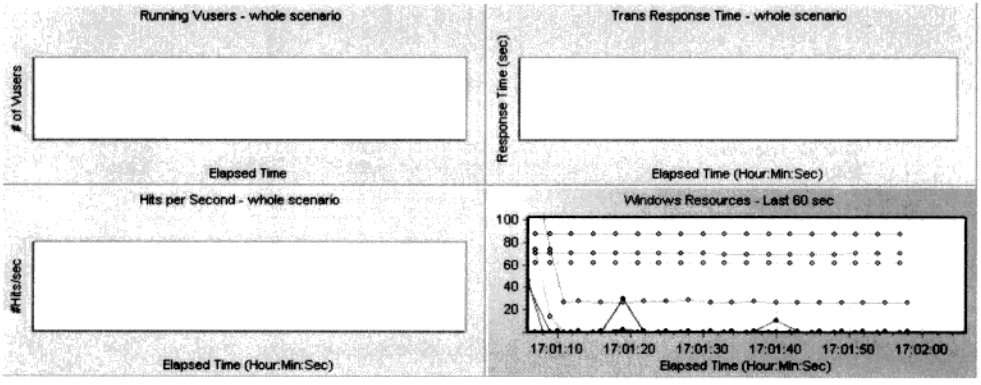


图 11-27 监视数据图的信息

11.6 结果分析

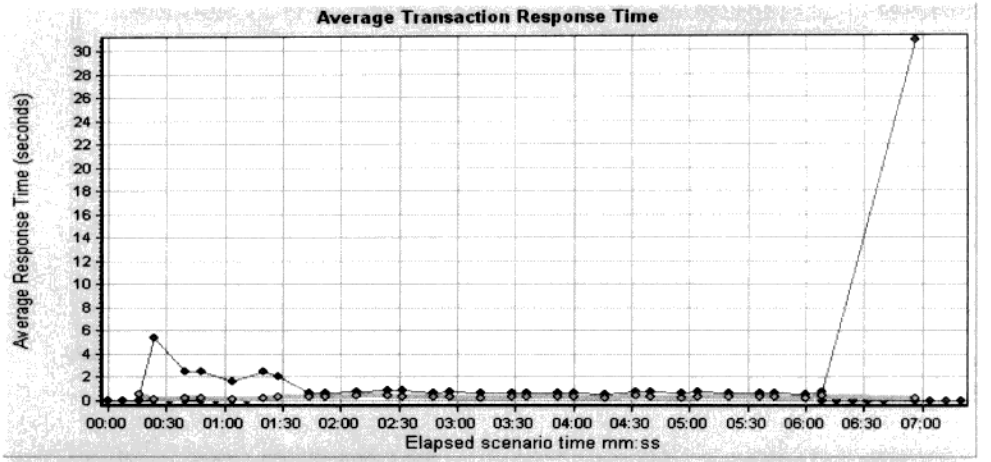
脚本执行完成后，分析测试结果，各模块测试结果分析如下：

1. 登录

首先分析平均事务响应时间图，如图 11-28 所示。

从图中不难看出，事务响应的平均时间为 0.303 秒，并且平均事务响应时间的曲线图表现得较平衡。这说明服务器表现得很优秀，但有一个问题是，手动去操作一次登录的过程会发现从双击桌面快捷方式到弹出登录对话框，加上从登录对话框输入正确的用户名和密码进入系统主界面的时间

根本不只 0.303 秒，并且大部分时间是双击桌面快捷方式到弹出登录对话框的时间，从登录对话框到系统主界面的时间也很短。接着分析测试脚本。



Color	Scale	Measurement	Graph's Mini...	Graph's Ave...	Graph's Max...	Graph's Me
☑	1	Action_Transaction	0.519	2.105	30.809	0.711
☑	1	vuser_end_Transaction	0	0	0	0
☑	1	vuser_init_Transaction	0.001	0.001	0.001	0.001
☑	1	登录	0.153	0.303	0.504	0.297

图 11-28　平均事务响应时间图

```
soap_request("StepName=GetSystemInfo",
            "URL=http://192.168.8.56/test/SysSet.asmx",
            "SOAPEnvelope=<?xml version=\"1.0\" encoding=\"GBK\" standalone=\"no\"?"
            "><soap:Envelope xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\" xmlns"
            ":xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xmlns:soap=\"http:/"
            "/schemas.xmlsoap.org/soap/envelope/\"><soap:Body><GetSystemInfo xmlns="
            "\"http://tempuri.org/\"/></soap:Body></soap:Envelope>",
            "Snapshot=t1.inf",
            "ResponseParam=response",
            LAST);
        lr_think_time(12);
        web_add_header("Content-Type", "text/xml; charset=utf-8");
        web_add_header("SOAPAction", "\"http://tempuri.org/IsIpValid\"");
        soap_request("StepName=IsIpValid",
            "URL=http://192.168.8.56/test/ULogin.asmx",
            "SOAPEnvelope=<?xml version=\"1.0\" encoding=\"GBK\" standalone=\"no\"?"
            "><soap:Envelope xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\" xmlns"
```

```
":xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xmlns:soap=\"http:/"
"/schemas.xmlsoap.org/soap/envelope/\"><soap:Body><IsIpValid xmlns=\""
"http://tempuri.org/\"/></soap:Body></soap:Envelope>",
"Snapshot=t2.inf",
"ResponseParam=response",
LAST);
```

从脚本中可以看出，当双击桌面快捷方式后，系统调用的是 http://192.168.8.56/test/SysSet.asmx 文件，而弹出登录对话框后，系统调用的是 http://192.168.8.56/test/ULogin.asmx 文件，SysSet.asmx 是产品注册管理的文件，ULogin.asmx 是用户管理文件，并且从脚本中还可以看到 think_time 的时间为 12 秒，这就是为什么双击桌面快捷方式后，没有立即弹出登录对话框的原因，那么看到的平均事务的响应时间为什么那么短呢？这是因为在进行分析的时候没有将 think_time 的时间考虑进去，现在通过页面细分来分析各页面执行所花的时间，如图 11-29 所示。

Legend

Color	Scale	Measurement	Minimum	Average	Maximum
☑	1	192.168....emSetting.asmx [mai...	0.053	0.14	0.363
☐	1	192.16.../WsUcSearch.asmx [...	0	0.007	0.036
☐	1	192.16...s/WsUcLogin.asmx [...	0	0.143	0.568
☐	1	192.168....anagement.asmx [m...	0	0.012	0.07

图 11-29 页面细分图

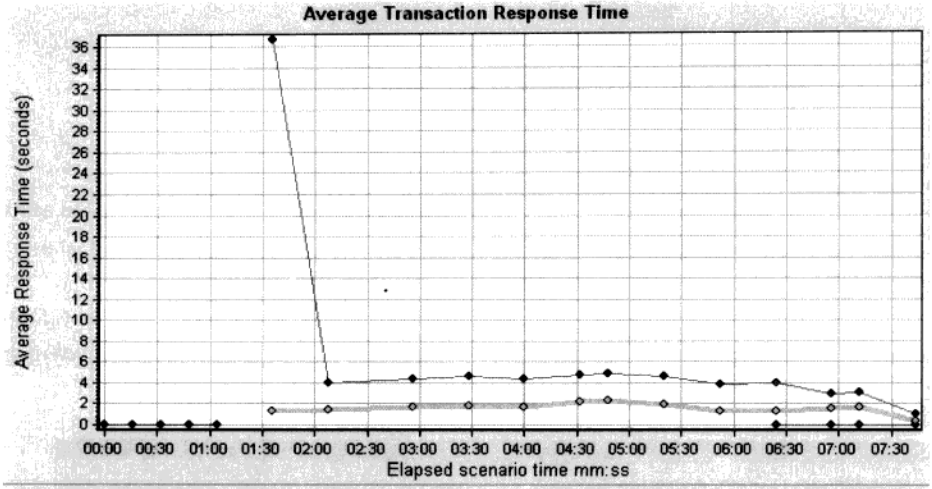
从上图中可以看到这几个文件执行的时间都很短，所以可以说明单个的这些 asmx 文件是没有问题的。

现在整个登录过程最大的问题在于双击桌面快捷方式到弹出登录对话框所花的时间太长。接着分析产品注册管理页面（http://192.168.8.56/test/SysSet.asmx），提交请求后系统在做些什么？在这段时间系统做了两件事：注册产品信息验证和加载所有的服务。注册产品信息验证必须和数据库打交道，需要到数据库中去核实产品信息是否正确，这是一个查询的过程。为了得到注册产品信息所花的时间，在这里使用 SQL Server 2005 的 Dashboard 进行性能监控，但监控过程中并未发现数据库存在性能的问题。这样问题最有可能是由于加载服务时引起的，当然加载这些服务肯定是要时间的，这点不用怀疑，但是否需要那么长的时间呢，这个不得而知，为了验证加载服务的时间长短，开发工程师对程序进行了一些修改，即在双击桌面快捷方式时，系统不加载这些服务，而是直接弹出登录对话框，结果发现整个登录过程节约了 5 秒左右的时间。

通过上面的实验得到的结论是：登录的时间过长是由登录过程中必须加载所有的服务引起的。对于 C/S 模式架构的系统，加载服务的过程都是在客户端完成，如果不能对加载服务的过程进行处理，就无法对登录的时间进行调优。

2. 数据检索

场景运行完成后，首先分析平均事务响应时间，如图 11-30 所示。



Color	Scale	Measurement	Graph's Mini...	Graph's Ave...	Graph's Max...	Graph's Me
☑	1	Action_Transaction	0.974	6.411	36.66	4.41
☑	1	vuser_end_Transaction	0	0	0	0
☑	1	vuser_init_Transaction	0.001	0.001	0.001	0.001
☑	1	检索	0.375	1.582	2.35	1.61

图 11-30　平均事务响应时间图

从图中可以看出，平均时间为 1.582 秒，这说明系统检索 100 条记录是相当得快，并且在整个检索过程中没有出现失败的事务，所有事务都全部成功，系统性能表现得很优秀。

3. 进入新增记录界面

场景运行完成后，首先分析平均事务响应时间，如图 11-31 所示。

图中所示的平均时间为 0.161 秒，这个时间表现应该是很优秀，但是手动操作进入新增记录界面发现，进入新增记录界面的时间大概为 1～2 秒左右。那么为什么测试出来的时间会有这么大的差距呢？经过和开发工程师沟通才知道了原因，因为在点击“新增记录”按钮时，系统会对弹出的新增记录对话框中的控件进行初始化，而这部分时间并没有被计算到进入新增记录界面事务的时间内，而是被计算在 systemsetting 的页面内，所以在测试过程中，无法监控到这段时间，故结果和实际的有点差距，大概在 0.5 秒左右。

4. 提交新增记录

场景运行完成后，首先分析提交新增记录的平均事务响应时间，如图 11-32 所示。

从图中可以看出，提交新增记录的平均事务响应时间为 1.851 秒，这个时间表现应该说很不错，但是从图中不难发现，平均事务响应时间图是一个波浪形的，也就是说平均事务响应时间图中那个方差值太大。这说明可能是应用服务器出现问题，在场景执行过程中，应用服务器可能无法处理客户端的请求。

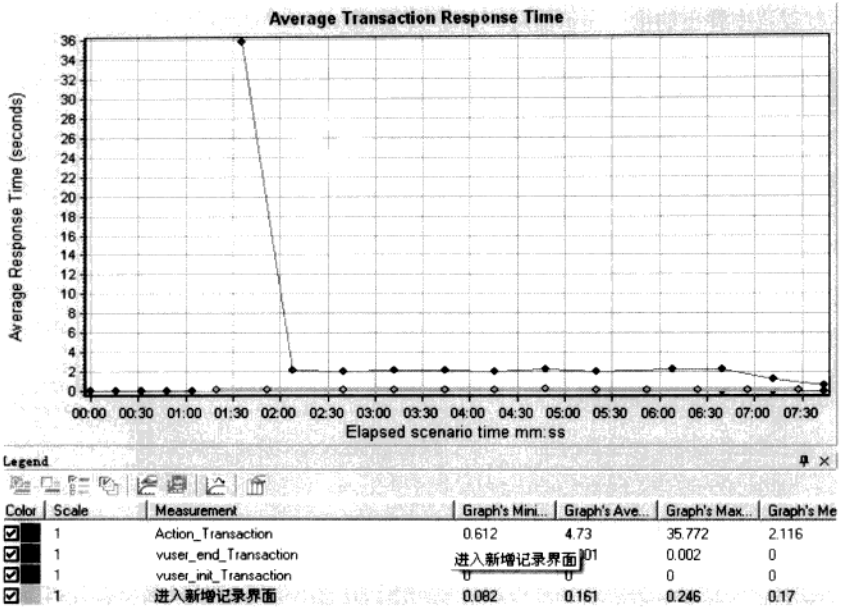


图 11-31 进入新增记录界面平均事务响应时间图

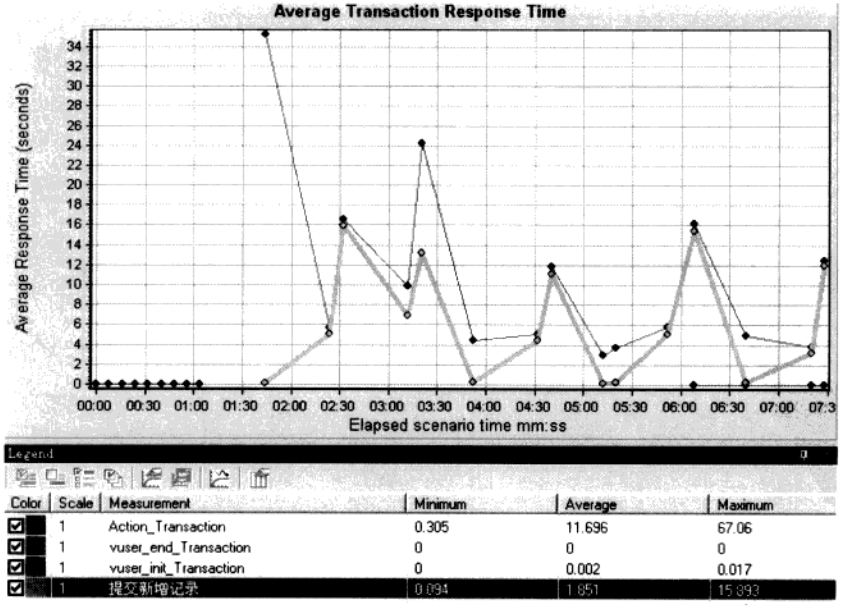


图 11-32 提交新增记录平均事务响应图

接着分析场景运行过程中弹出的错误提示信息，如图 11-33 所示。

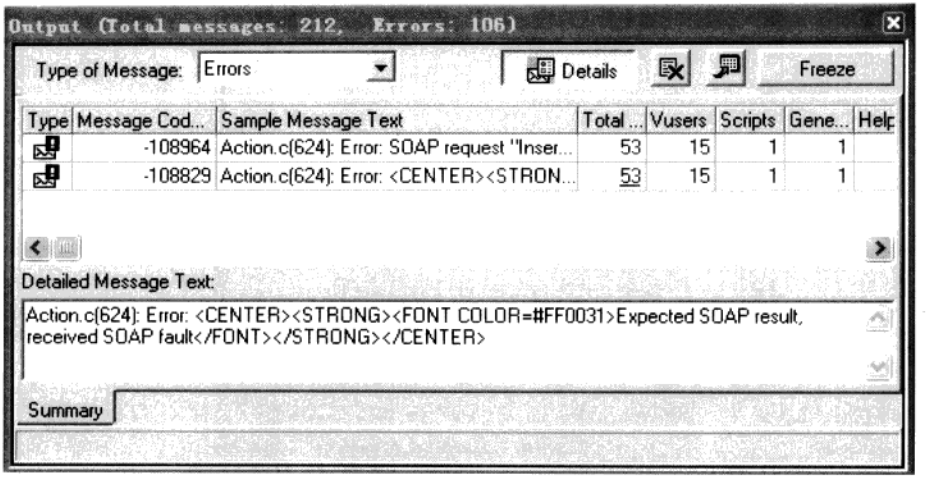


图 11-33　错误提示信息

从上图的输出对话框中，可以看到当脚本运行到第 624 行时失败。找到虚拟用户运行的日志，如图 11-34 所示。

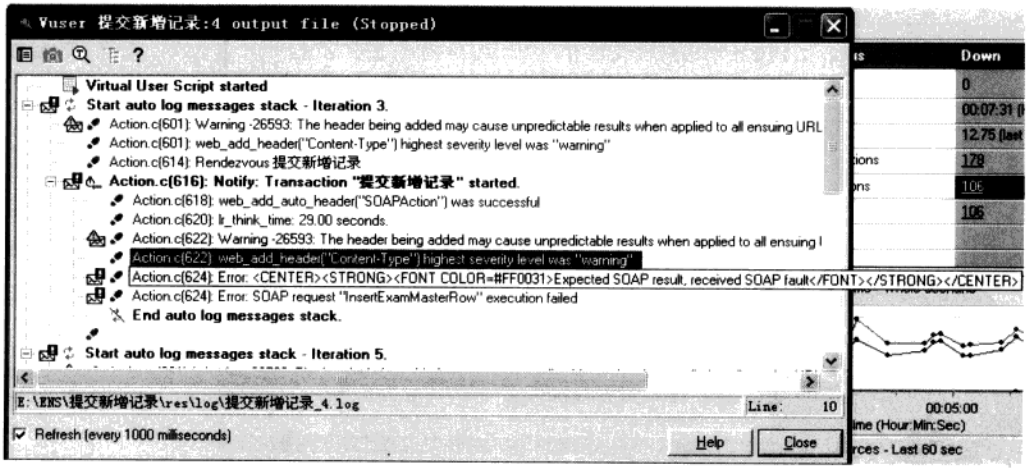


图 11-34　虚拟用户日志文件

从虚拟用户运行时的日志文件中，可以很清楚的看到，在脚本迭代的过程中，有部分迭代失败，并且每个虚拟用户都出现这种情况。那么脚本中第 624 行代码主要是用来完成哪些动作的呢？分析下面这段代码。

```
soap_request("StepName=InsertExamMasterRow",
        "URL=http://192.168.8.56/test/Exam.asmx",
        "SOAPEnvelope=<?xml version=\"1.0\" encoding=\"GBK\" standalone=\"no\"?"
        "><soap:Envelope xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\" xmlns"
        ":xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xmlns:soap=\"http:/"
        "/schemas.xmlsoap.org/soap/envelope/\"><soap:Body><InsertExamMasterRow "
        "xmlns=\"http://tempuri.org/\"><appoint_id/><performed_by>1</"
        "performed_by><exam_class>1</exam_class><exam_item>1</exam_item>"
        "<patient_id/><patient_name>{recrod}</patient_name><sex>O</sex>"
        "<date_of_birth xsi:nil=\"true\"/><race>MONLIAN</race><unit/>"
        "<mailing_address/><zip_code/><phone_number/><age xsi:nil=\"true\"/>"
        "<body_height xsi:nil=\"true\"/><body_weight xsi:nil=\"true\"/>"
        "<systolic_pressure xsi:nil=\"true\"/><diastolic_pressure xsi:nil=\""
        "true\"/><exam_reason/><clin_symp/><phys_sign/><relevant_lab_test/>"
        "<relevant_diag/><patient_source>1</patient_source><visit_id/><ward_no/"
        "><sickbed_no/><req_date_time xsi:nil=\"true\"/><req_dept_name/>"
        "<req_physician/><req_memo/><device_id>1</device_id><result_status>1</"
        "result_status><custom_1/><custom_2/><medicine_history/>"
        "<patient_statement/><disease_history/><regist_user_id>test1</"
        "regist_user_id><regist_date_time>2009-12-21T17:44:26.3086475+08:00</"
        "regist_date_time></InsertExamMasterRow></soap:Body></soap:Envelope>",
        "Snapshot=t40.inf",
        "ResponseParam=response",
        LAST);
```

从这段代码中可以看到，这个过程是将填写好的记录信息进行提交的过程，如果还是不明确，可以分析http://192.168.8.56/test/Exam.asmx?op=InsertExamMasterRow页面内容，该页面的源代码写的很清楚。

经过上面的分析，可以很明确的看到，平均事务响应时间之所以是一条波浪形的曲线是因为在脚本迭代的过程中，系统无法响应部分提交新增记录的信息的请求，导致整个请求的时间过长，最后失败。

当然应用服务器的处理能力还会受到其他因素的影响，首先分析脚本运行时的系统资源的表现情况，通过分析系统资源图发现在整个脚本运行的过程中，系统资源表现正常，没有任何问题。同时通过监控数据库的表现，发现数据库也很正常，这说明应用服务器的处理能力并未受到其他条件的影响。

接下来可以调优应用服务器去解决这个问题，否则系统就无法支持 15 个虚拟用户并发提交新增记录。

当然从上面的分析也可以看到，只要成功提交新增记录，平均事务响应时间都在 1～2 秒内，这说明数据库在处理新增记录时没有任何问题；接下来分析每个虚拟用户对应的提交新增记录的平均事务响应时间的情况。

在平均事务响应时间图中，右击选择 Drill Down，在钻取设置对话框中的 Group By 中选择 VuserID，如图 11-35 所示。

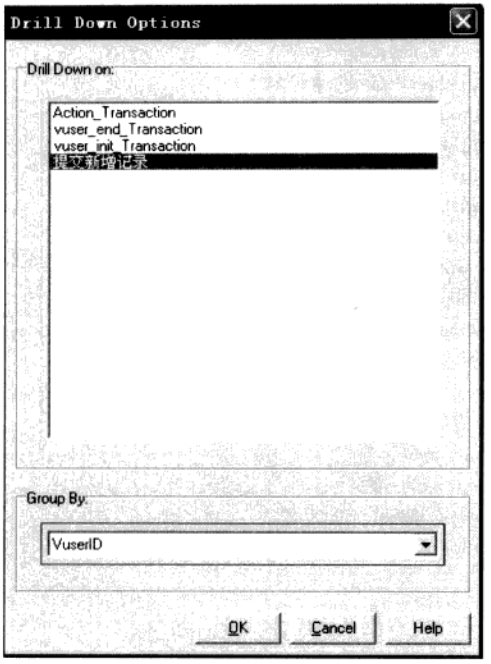
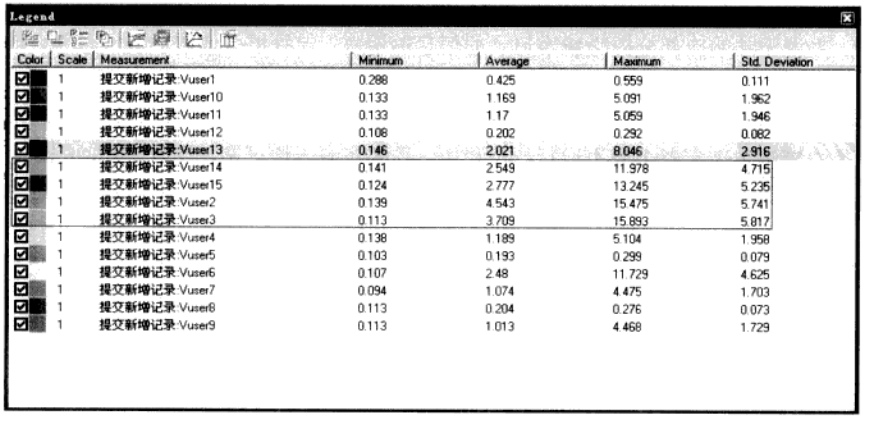


图 11-35 钻取设置

每个虚拟用户对应的平均事务响应时间图，如图 11-36 所示。



Color	Scale	Measurement	Minimum	Average	Maximum	Std. Deviation
☑	1	提交新增记录:Vuser1	0.288	0.425	0.559	0.111
☑	1	提交新增记录:Vuser10	0.133	1.169	5.091	1.962
☑	1	提交新增记录:Vuser11	0.133	1.17	5.059	1.946
☑	1	提交新增记录:Vuser12	0.108	0.202	0.292	0.082
☑	1	提交新增记录:Vuser13	0.146	2.021	8.046	2.916
☑	1	提交新增记录:Vuser14	0.141	2.549	11.978	4.715
☑	1	提交新增记录:Vuser15	0.124	2.777	13.245	5.235
☑	1	提交新增记录:Vuser2	0.139	4.543	15.475	5.741
☑	1	提交新增记录:Vuser3	0.113	3.709	15.893	5.817
☑	1	提交新增记录:Vuser4	0.138	1.189	5.104	1.958
☑	1	提交新增记录:Vuser5	0.103	0.193	0.299	0.079
☑	1	提交新增记录:Vuser6	0.107	2.48	11.729	4.625
☑	1	提交新增记录:Vuser7	0.094	1.074	4.475	1.703
☑	1	提交新增记录:Vuser8	0.113	0.204	0.276	0.073
☑	1	提交新增记录:Vuser9	0.113	1.013	4.468	1.729

图 11-36 每个虚拟用户对应的平均事务响应时间图

从图中可以看出，每个虚拟用户提交新增信息的平均时间都小于 5 秒，这还包括失败的请求，因为一般失败的请求肯定会超过 5 秒，这样失败请示的时间其实是将平均值拉大了，成功的请求平均时间一定会小于 5 秒，故只要解决应用服务器处理请求的问题，提交新增记录的速度一定能满足需求，当然这里只是针对 15 个虚拟用户的情况，如果有几百个虚拟用户就要另当别论了。图中还有 4 个虚拟用户的最大时间超过了 10 秒。

最后经过调优应用服务器解决了该问题，重新测试平均事务响应时间为 1.521 秒。

11.7 测试结论

测试结果分析完成后，本节对上面每个模块的测试结果进行简单的总结：

1. 登录

登录模块，整个登录过程中调用到的 asmx 文件性能都没有任何问题，数据库性能也没有任何问题，时间过长是由于在登录过程中必须加载所有服务并对这些服务进行初始化引起的。

2. 数据检索

系统在 50 万条记录中检索出 100 条记录的平均时间为 1.582 秒，系统性能表现优秀。

3. 新入新增记录界面

进入新增记录界面的时间表现达到性能要求，但是测试到的时间比起实际的时间短了 0.5 秒左右，这是因为在点击“新增记录”时，系统对弹出的新增记录对话框的控件进行了初始化，而这部分时间被计算到 systemsetting 页面内，故测试到的时候比起实际要小一点，但整个功能的性能达到要求。

4. 提交新增记录

系统无法支持 15 个虚拟用户并发提交新增记录，主要原因是应用服务器无法处理 15 个虚拟用户并发，经过调优应用服务器，解决该问题后平均事务响应时间达到要求，平均事务响应时间为 1.521 秒。

A

主要计数器

在这部分主要整理了一些计数器及阈值要求，Windows 计数器见表 A-1。

表 A-1　Windows 计数器

Object （对象）	Counters （计数器名称）	Description （描述）	参考值
Memory	Available Mbytes	可用物理内存数。如果 Available Mbytes 的值很小（4 MB 或更小），则说明计算机上总的内存可能不足，或某程序没有释放内存。每个附加连接将在此基础上占用 10KB 左右	至少要有 10%的物理内存值
Memory	page/sec	表示因为页面错误，从磁盘取出的页面数，或是由于页面错误，写入磁盘以释放工作空间的页面数	page/sec 推荐值为 0～20，一般如果 pages/sec 持续高于几百，那么应该进一步研究页交换活动
Memory	page read/sec	每秒读取的页面数	阈值为 5，越低越好，阈值大表示是从磁盘读而不是从缓存中读
Memory	Page Faults/sec	每秒失效页面数（包括软失效和硬失败）	
Memory	Cache Bytes	文件系统缓存（File System Cache）	默认情况下为 50%的可用物理内存。如 IIS5.1 运行内存不够时，它会自动整理缓存。需要关注该计数器的趋势变化
Memory	Pages per second	每秒钟检索的页数	应该少于每秒一页
Process	%Processor Time	CPU 使用率，查看处理器饱和状态	小于 75%

续表

Object（对象）	Counters（计数器名称）	Description（描述）	参考值
Process	%User Time	表示耗费 CPU 的数据库操作，如排序、执行、aggregate functions 等	如果该值很高，可考虑增加索引，尽量使用简单的表联接，水平分割大表格等方法来降低该值
Process	%Privileged Time	（CPU 内核时间）是在特权模式下处理线程执行代码所花时间的百分比	如果该参数值和 Physical Disk 参数值一直很高，表明 I/O 有问题。可考虑更换更快的硬盘系统
Process	Processor Queue Length	处理器队列的瞬时长度，以线程数为单位	
Process	DPC Time	越低越好。在多处理器系统中，如果这个值大于 50%并且 Processor：% Processor Time 非常高，加入一个网卡可能会提高性能，提供的网络已经不饱和	判断 CPU 瓶颈，如果 processor queue length 显示的队列长度保持不变（>=2）并且处理器的利用率%Processor time 超过 90%，那么很可能存在处理器瓶颈
Thread	ContextSwitches/sec	实例化 inetinfo 和 dllhost 进程，如果你决定要增加线程字节池的大小，应该监视这三个计数器。增加线程数可能会增加上下文切换次数，这样性能不会上升反而会下降。如果 10 个实例的上下文切换值非常高，就应该减小线程字节池的大小	
Physical Disk	% Disk Time	指所选磁盘驱动器忙于为读取或写入请求提供服务所用的时间的百分比	正常值<10，此值过大表示耗费太多时间来访问磁盘，可考虑增加内存、更换更快的硬盘、优化读写数据的算法
Physical Disk	Avg.Disk Queue Length	指读取和写入请求为所选磁盘在实例间隔中列队的平均数	正常值<0.5，此值过大表示磁盘 I/O 太慢，要更换更快的硬盘
Physical Disk	Current Disk Queue Length	收集性能数据时磁盘上当前的请求数量，它还包括在收集时处于服务的请求。这是瞬间的快照，不是时间间隔的平均值。多轴磁盘设备能有一次处于运行状态的多重请求，但是其他同期请求正在等待服务	

监视 SQL Server 的计数器见表 A-2。

表 A-2 SQL Server 计数器

Object（对象）	Counters（计数器名称）	Description（描述）	参考值
SQLServer	Full Scans/sec	（全表扫描/秒）每秒不受限的完全扫描数，可以是基本表扫描或全索引扫描	如果这个计数器显示的值比 1 或 2 高，应该分析你的查询以确定是否确实需要全表扫描，以及 SQL 查询是否可以被优化
SQLServer	Page splits/sec	（页分割/秒）由于数据更新操作引起的每秒页分割的数量	
SQLServer	Page Reads/sec	每秒发出的物理数据库页读取数。这一统计信息显示的是在所有数据库间的物理页读取总数	由于物理 I/O 的开销大，可以通过使用更大的数据高速缓存、智能索引、更高效的查询或者改变数据库设计等方法，使开销减到最小
SQLServer	Page Writes/sec	（写的页/秒）每秒执行的物理数据库写的页数	
SQLServer	Buffer Cache Hit Ratio	在“缓冲池”（Buffer Cache/Buffer Pool）中没有被读过的页占整个缓冲池中所有页的比率。可在高速缓存中找到而不需要从磁盘中读取的页的百分比	这一比率是高速缓存命中总数除以自 SQL Server 实例启动后对高速缓存的查找总数。经过很长时间后，这一比率的变化很小
SQLServer	Lazy Writes/sec	（惰性写/秒）惰性写进程每秒写的缓冲区的数量	值最好为 0
SQLServer	Cache Hit Ratio	高速缓存命中次数和查找次数的比率	
SQLServer	Average Latch Wait Time(ms)	（平均闩等待时间（毫秒））一个 SQL Server 线程必须等待一个闩的平均时间，以毫秒为单位	
SQLServer	Latch Waits/sec	（闩等待/秒）在闩上每秒的等待数量	如果这个值很高，表明你正经历对资源的大量竞争
SQLServer	Number of Deadlocks/sec	（死锁的数量/秒）导致死锁的锁请求的数量	
SQLServer	Average Wait Time(ms)	（平均等待时间（毫秒））线程等待某种类型的锁的平均等待时间	
SQLServer	Lock Requests/sec	（锁请求/秒）每秒钟某种类型的锁请求的数量	

监视 IIS 的计数器见表 A-3。

表 A-3 IIS 计数器

Object（对象）	Counters（计数器名称）	Description（描述）	参考值
IIS	File Cache Hits %	全部缓存请求中缓存命中次数所占的比例，反映了 IIS 的文件缓存设置的工作情况	对于一个大部分是静态网页组成的网站，该值应该保持在 80%左右
IIS	Bytes Total/sec	显示 Web 服务器发送和接收的总字节数，低数值表明该 IIS 正在以较低的速度进行数据传输	
IIS	Connection Refused	数值越低越好，高数值表明网络适配器或处理器存在瓶颈	
IIS	Not Found Error	显示由于被请求文件无法找到而无法由服务器满足的请求数	

B

性能测试 i 模型

在这部分介绍一个性能测试模型，该模型称之为 i 模型，i 模型如图 B-1 所示。

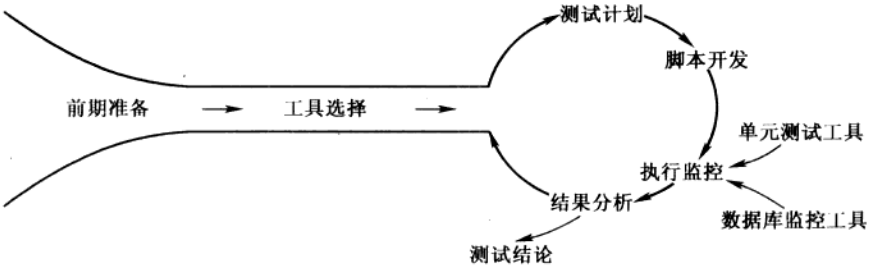


图 B-1　性能测试 i 模型

这个模型是个人对性能测试过程的一个理解，仅供大家参考。该模型之所以叫 i 模型，是因为这个模型长的比较像“i”字母。

i 模型性能测试过程中各阶段完成的工作如下：

1. 前期准备阶段

在这个阶段主要是为即将开展的性能测试，搭建一支合适的性能测试队伍。这点与黑盒测试有点不同，黑盒测试很少有这样的一个过程，之所以需要这个过程是因为性能测试需要较多的技术和技能，而正常情况下单个性能测试工程师不太可能都很熟悉这些技术和技能，所以在性能结果分析的过程中，就需要一个团队来支持。一般一个性能测试团队应该包括：项目测试经理、测试设计工程师、测试开发工程师、测试执行工程师、测试分析工程师、系统管理员、网络工程师和 DBA（数据库工程师）。

各角色的职责见表 B-1。

表 B-1 性能测试团队角色及职责

角色	职责
测试经理	1. 和项目组成员交流，确定测试需求和测试环境 2. 制定测试计划 3. 控制测试进度 4. 风险管理
测试设计工程师	1. 制定性能测试方案 2. 定义性能测试范围 3. 确定性能测试业务模型 4. 确定性能测试场景模型
测试开发工程师	1. 脚本开发、调试 2. 确定性能场景 3. 确定需要监控的性能指标和计数器
测试执行工程师	1. 搭建测试环境 2. 准备测试数据 3. 执行测试脚本 4. 记录测试结果
测试分析工程师	1. 根据测试结果进行分析 2. 找到性能瓶颈并提出调优建议
系统管理员	系统支持，解决测试工程师无法解决的系统问题
网络工程师	提供网络方面的支持，需要时可以就网络方面进行分析和支持
DBA	对数据库方面的性能分析提供支持

但很多情况下项目测试经理、测试设计工程师、测试开发工程师、测试执行工程师、测试分析工程师是由同一个人担任，特别是一些比较小的项目，较大的项目可能会分得比较细。

2. 工具选择

在进行性能测试之前还必须确定使用的性能测试工具，这就是工具选择的过程，有的时候甚至还需要一个工具选型的报告。当确定好测试工具后，有时还需要就该测试工具的相关知识对项目组成员进行培训。

3. 测试计划

上面的工作准备完成后，便可着手进行性能测试活动了。在测试计划阶段需要完成测试计划、测试方案和风险分析。这点和功能测试没有区别。在测试方案部分必须明确性能测试的目标、业务模型和场景模型，否则接下来的脚本开发可能会有麻烦。

4. 脚本开发

脚本开发过程中，需要依据测试方案来完成测试用例，并开发出符合业务模型和场景模型的测试脚本。在脚本中必须明确需要监控的指标或事务。

5. 执行监控

脚本开发完成后，接下来测试执行工程师开始执行脚本，在测试执行工程师执行脚本之前，还必须去搭建测试环境和准备测试过程中需要的数据。在脚本执行过程中，测试执行工程师必须对测试过程进行监控，并将测试结果进行记录，包括一些出错的信息，以及计数器的值。

在模型图中可以看到这里使用到单元测试工具和数据库监控、调优的工具，为什么要涉及到这些工具呢？因为在使用 LoadRunner 进行性能测试时，经常遇到这样的情况：即使发现了问题，但还是无法就性能瓶颈进行更深层次的分析，或者说根本无法具体定位到代码行，这时候就需要单元测试工具来协助测试。假设有一个系统，在测试过程中很明确知道是内存泄漏的原因引起的，但是无法找到哪个模块或哪段代码出现了问题，这个时候就需要找一些专门监控内存泄漏的工具来辅助 LoadRunner 进行测试。

而数据库监控和调优的工具也一样，假设测试一个检索的功能，发现检索 20 个数据的时间很长，LoadRunner 测试时，只知道是数据库检索时间太长引起的，但并不知道每条 select 语句的执行时间，这样就很难进行调优了。

6. 结果分析

脚本执行完成后，就需要对测试结果进行分析，去寻找性能瓶颈的原因。

那么为什么测试计划、脚本开发、执行监控和结果分析是一个环型呢？因为在实际的测试过程中，往往需要进行多次性能测试才能找到性能的瓶颈或者找到调优的办法，而每次进行性能测试，都必须对测试计划和测试方案进行修改。

最后测试完成，需要输出一份测试报告。

以上就是性能测试 i 模型的整个过程。